EXHIBIT B

US010292011B2

(12) **United States Patent**
Johnson

(10) **Patent No.: US 10,292,011 B2**
(45) **Date of Patent: May 14, 2019**

(54) **SYSTEM AND METHOD FOR LOCATION BASED EXCHANGE NETWORK**

(71) Applicant: **William J. Johnson**, Flower Mound, TX (US)

(72) Inventor: **William J. Johnson**, Flower Mound, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **16/147,532**

(22) Filed: **Sep. 28, 2018**

(65) **Prior Publication Data**

US 2019/0037354 A1 Jan. 31, 2019

Related U.S. Application Data

(63) Continuation of application No. 15/218,039, filed on Jul. 24, 2016, now Pat. No. 10,111,034, which is a continuation of application No. 14/752,945, filed on Jun. 28, 2015, now Pat. No. 9,456,303, which is a continuation of application No. 13/972,125, filed on Aug. 21, 2013, now Pat. No. 9,078,095, which is a

(Continued)

(51) **Int. Cl.**

H04M 11/04	(2006.01)
H04M 3/42	(2006.01)
H04W 4/02	(2018.01)
H04H 20/16	(2008.01)
H04L 12/24	(2006.01)
H04L 12/26	(2006.01)
H04L 29/08	(2006.01)
H04W 40/20	(2009.01)
H04W 40/24	(2009.01)
H04W 64/00	(2009.01)
H04W 12/06	(2009.01)
G06Q 10/08	(2012.01)
G06Q 30/06	(2012.01)

(52) **U.S. Cl.**
CPC ***H04W 4/023*** (2013.01); ***H04H 20/16*** (2013.01); ***H04L 41/0816*** (2013.01); ***H04L 43/16*** (2013.01); ***H04L 67/104*** (2013.01); ***H04W 4/02*** (2013.01); ***H04W 12/06*** (2013.01); ***H04W 40/20*** (2013.01); ***H04W 40/244*** (2013.01); ***H04W 64/00*** (2013.01); *G06Q 10/087* (2013.01); *G06Q 10/0833* (2013.01); *G06Q 30/0633* (2013.01)

(58) **Field of Classification Search**
CPC H04W 4/02; H04W 64/00; H04W 4/023; H04W 12/06; H04W 40/20; H04W 40/244; H04W 92/18; H04W 4/21; H04W 4/50; H04W 4/80; H04W 68/005; H04W 88/02
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,427,115 B1 *	7/2002	Sekiyama	....	G01C 21/3688 340/990
2001/0022558 A1 *	9/2001	Karr, Jr.	....	G01S 1/026 342/450

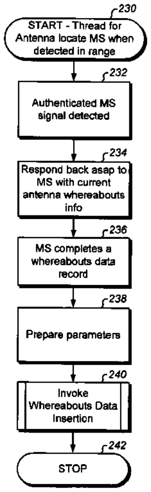
(Continued)

Primary Examiner — Liton Miah
(74) *Attorney, Agent, or Firm* — Yudell Isidore PLLC

(57) **ABSTRACT**

Mobile data processing Systems (MSs) interact with systems in their vicinity, and with each other, in communications and interoperability. Information transmitted inbound to, transmitted outbound from, is in process at, or is application modified at a mobile data processing system triggers processing of actions in accordance with user configurations, for example to present content to a user. The locatable network of MSs is referred to as a Location-Network Expanse.

20 Claims, 322 Drawing Sheets


230
START - Thread for Antenna locate MS when detected in range
232
Authenticated MS signal detected
234
Respond back asap to MS with current antenna whereabouts info
236
MS completes a whereabouts data record
238
Prepare parameters
240
Invoke Whereabouts Data Insertion
242
STOP

Related U.S. Application Data

continuation of application No. 12/590,831, filed on Nov. 13, 2009, now Pat. No. 8,634,796, which is a continuation-in-part of application No. 12/287,064, filed on Oct. 3, 2008, now Pat. No. 8,639,267, which is a continuation-in-part of application No. 12/077,041, filed on Mar. 14, 2008, now Pat. No. 8,600,341.

(56) References Cited

U.S. PATENT DOCUMENTS

2004/0252051	A1 *	12/2004	Johnson	G06F 17/3087 342/357.48
2007/0244633	A1 *	10/2007	Phillips	G08B 21/0236 701/408
2007/0281716	A1 *	12/2007	Altman	H04L 51/20 455/466

* cited by examiner

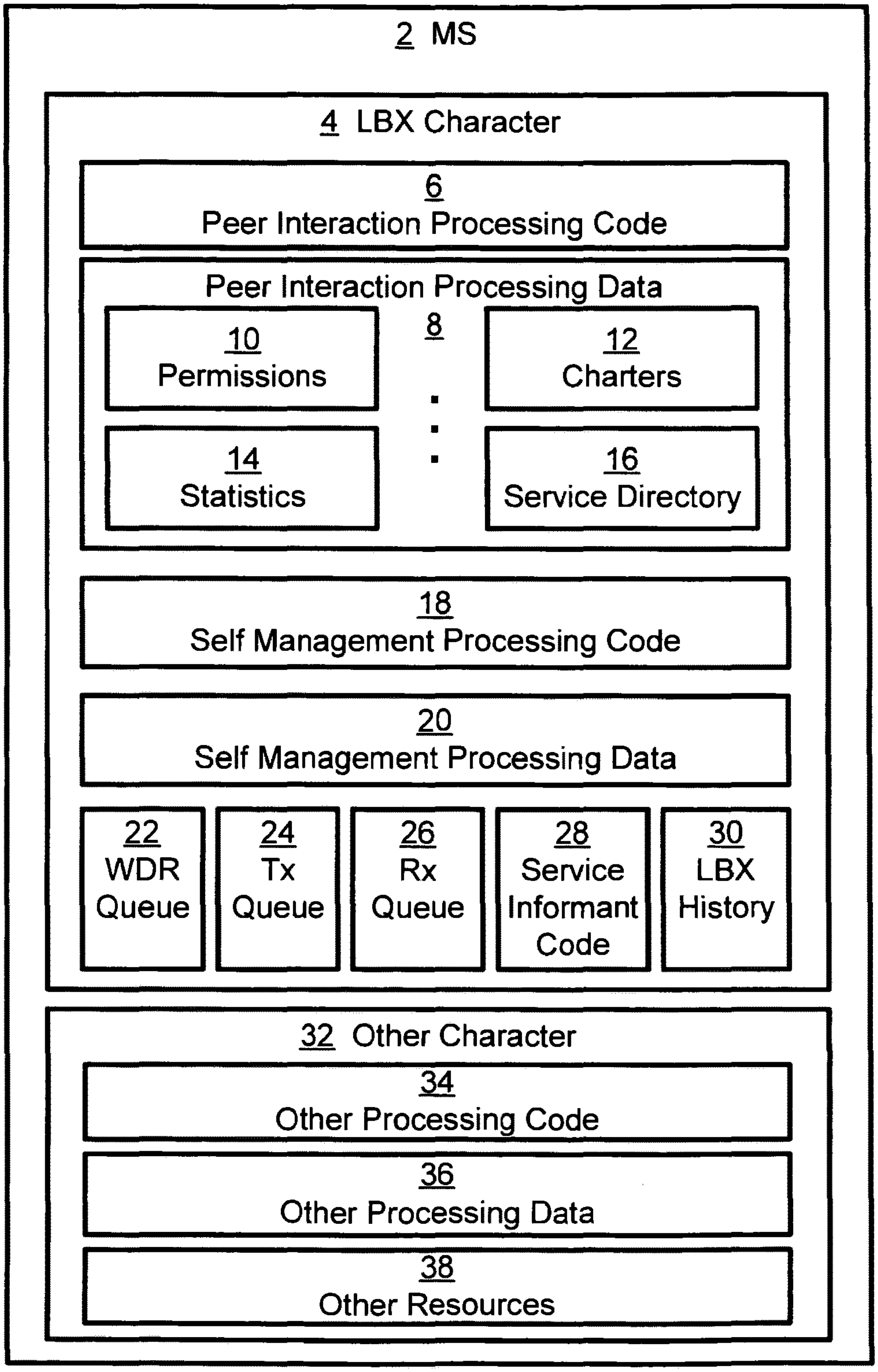

2 MS
4 LBX Character
6
Peer Interaction Processing Code
Peer Interaction Processing Data
8
10
Permissions
12
Charters
14
Statistics
16
Service Directory
18
Self Management Processing Code
20
Self Management Processing Data
22
WDR
Queue
24
Tx
Queue
26
Rx
Queue
28
Service
Informant
Code
30
LBX
History
32 Other Character
34
Other Processing Code
36
Other Processing Data
38
Other Resources

Fig. 1A

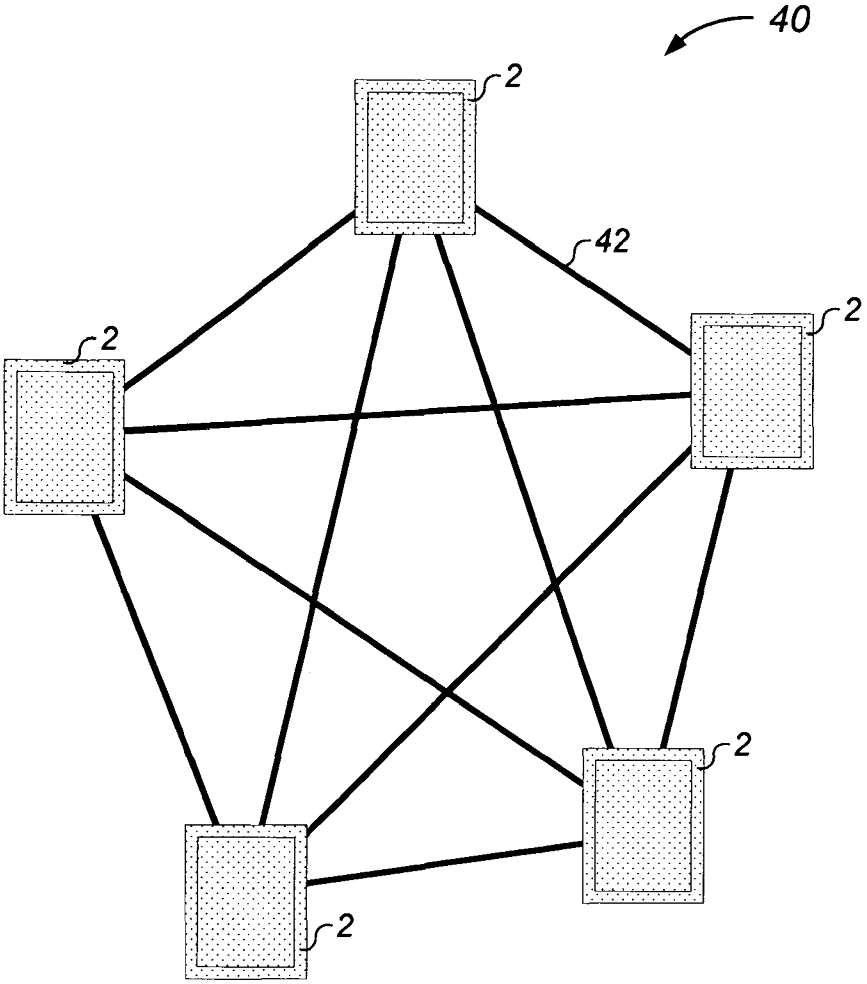


Fig. 1B

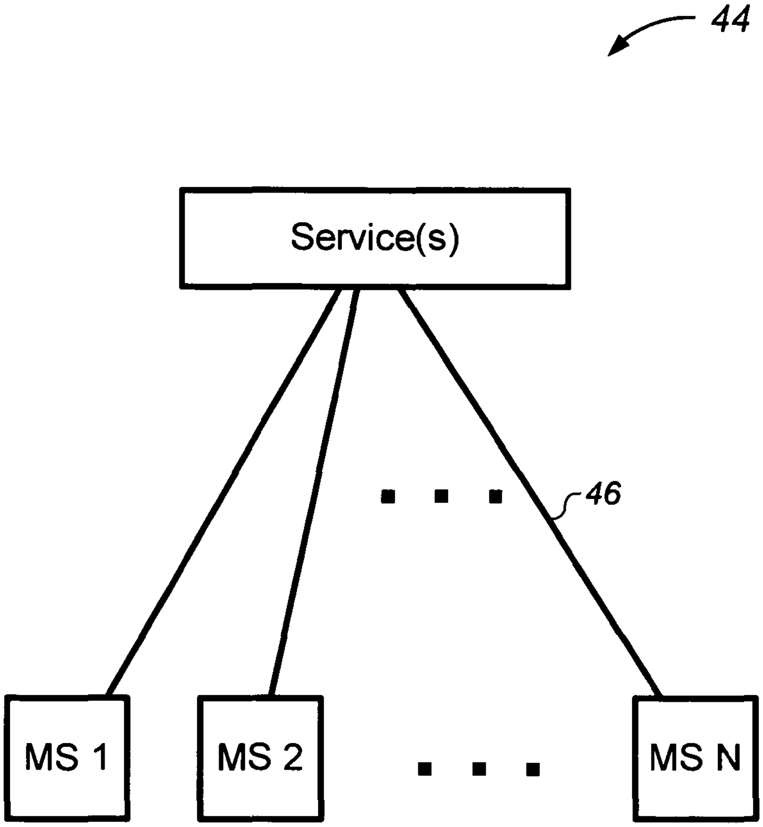


Fig. 1C

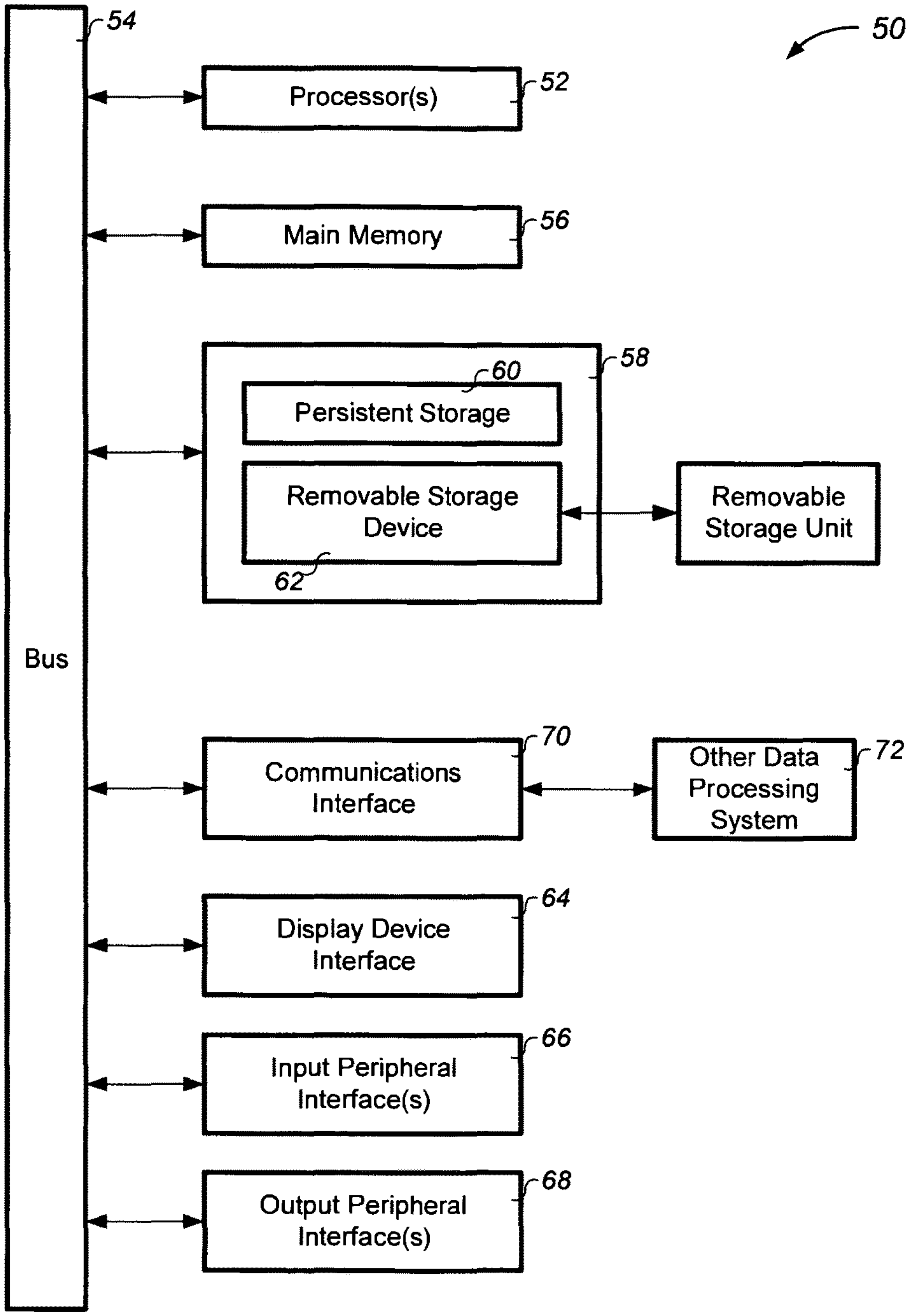

50
54
Bus
Processor(s)
52
Main Memory
56
58
60
Persistent Storage
Removable Storage Device
62
Removable Storage Unit
Communications Interface
70
Other Data Processing System
72
Display Device Interface
64
Input Peripheral Interface(s)
66
Output Peripheral Interface(s)
68

Fig. 1D

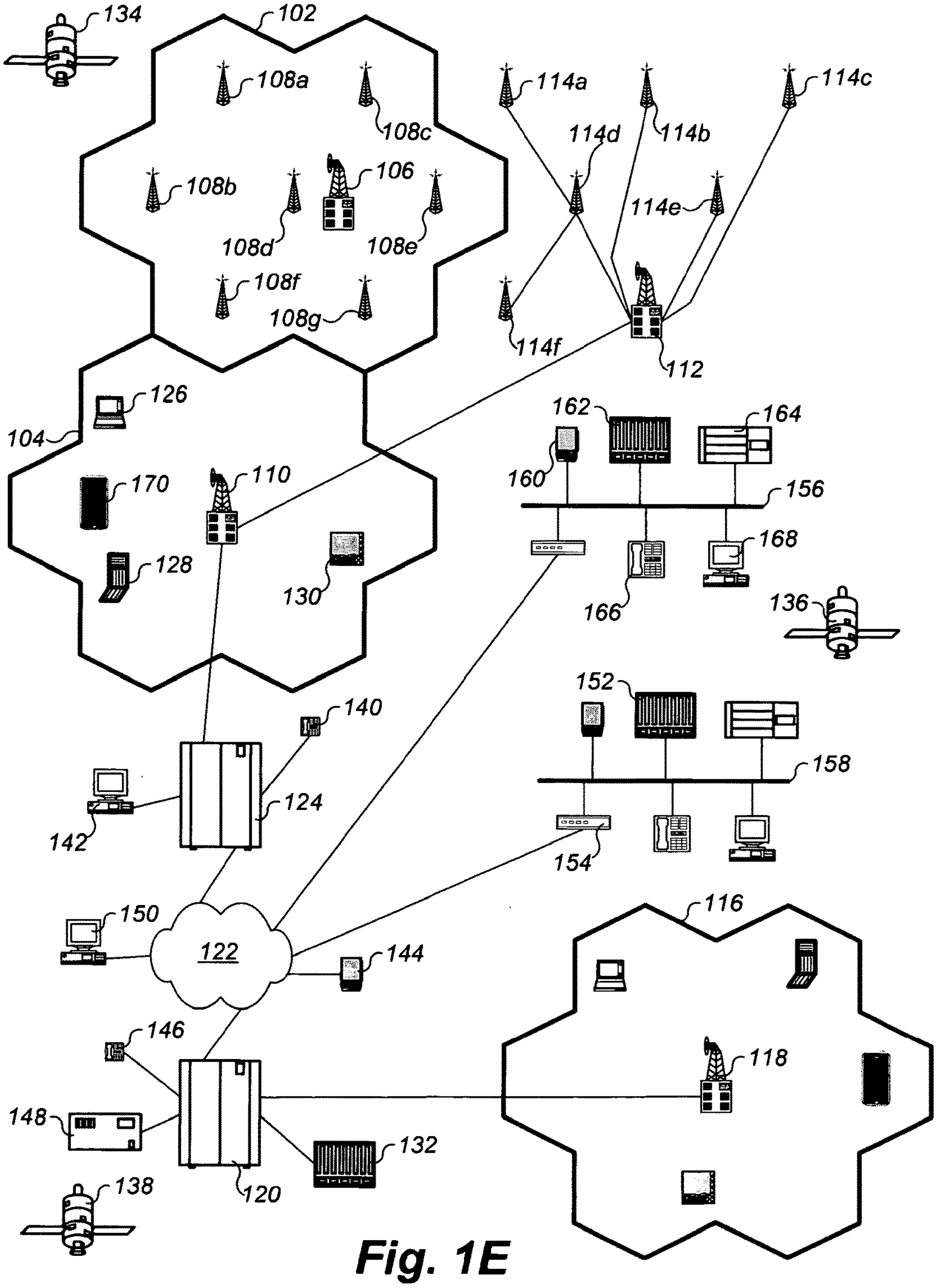

134
102
108a
108c
114a
114b
114c
114d
114e
114f
108b
106
108d
108e
108f
108g
112
126
104
170
110
128
130
162
164
160
156
168
166
136
152
158
140
124
142
154
150
122
144
116
146
118
148
132
138
120

Fig. 1E

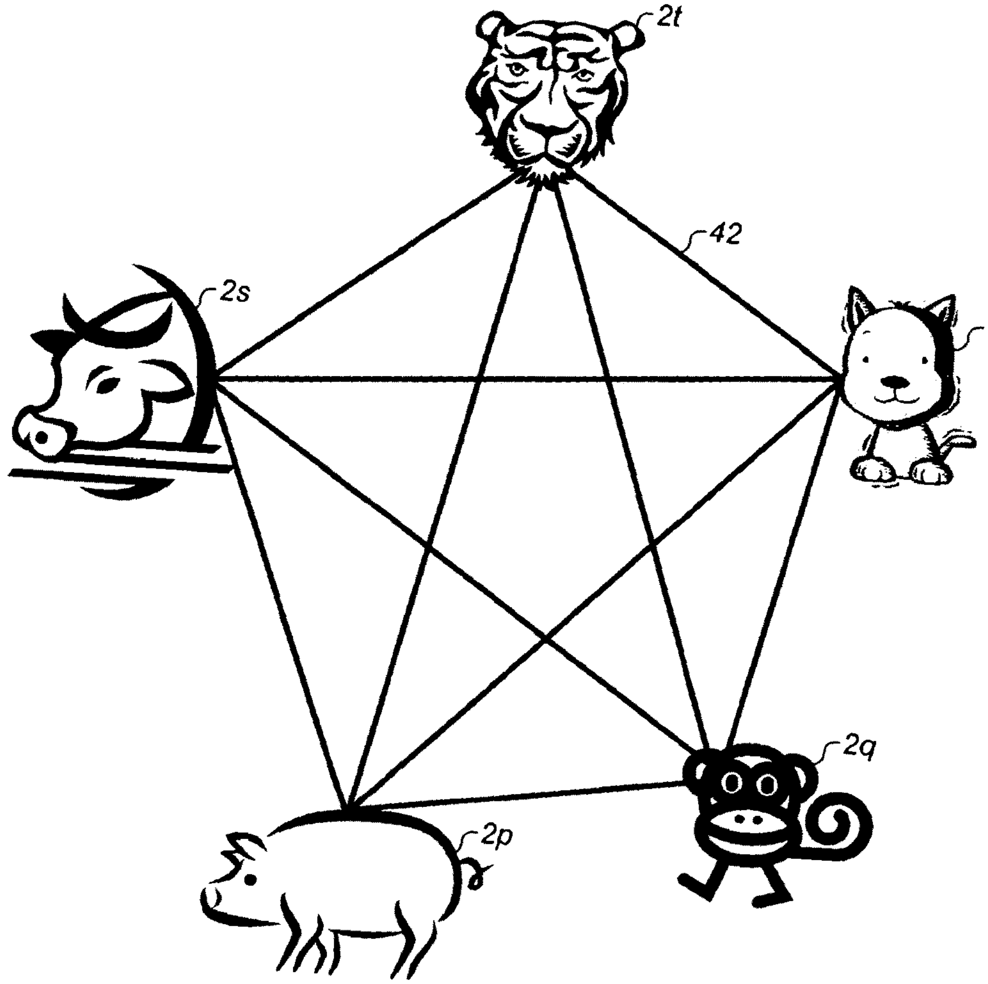


Fig. 1F

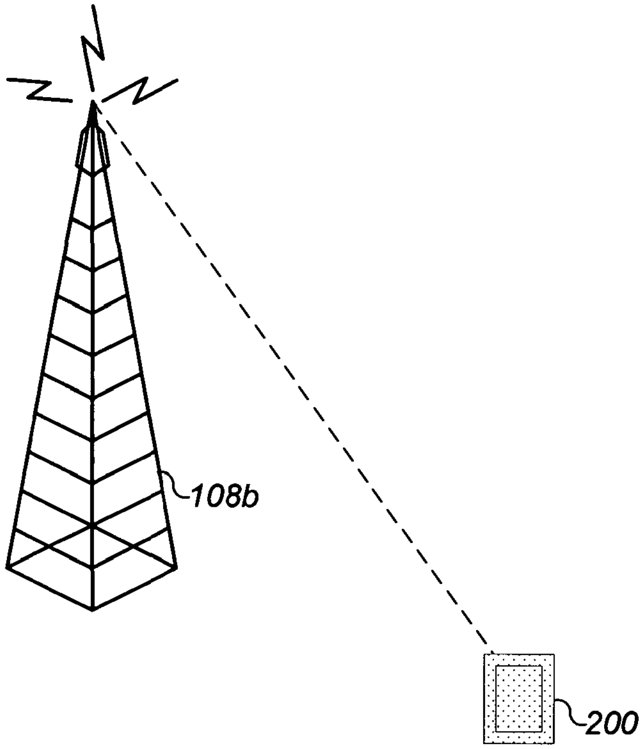


Fig. 2A

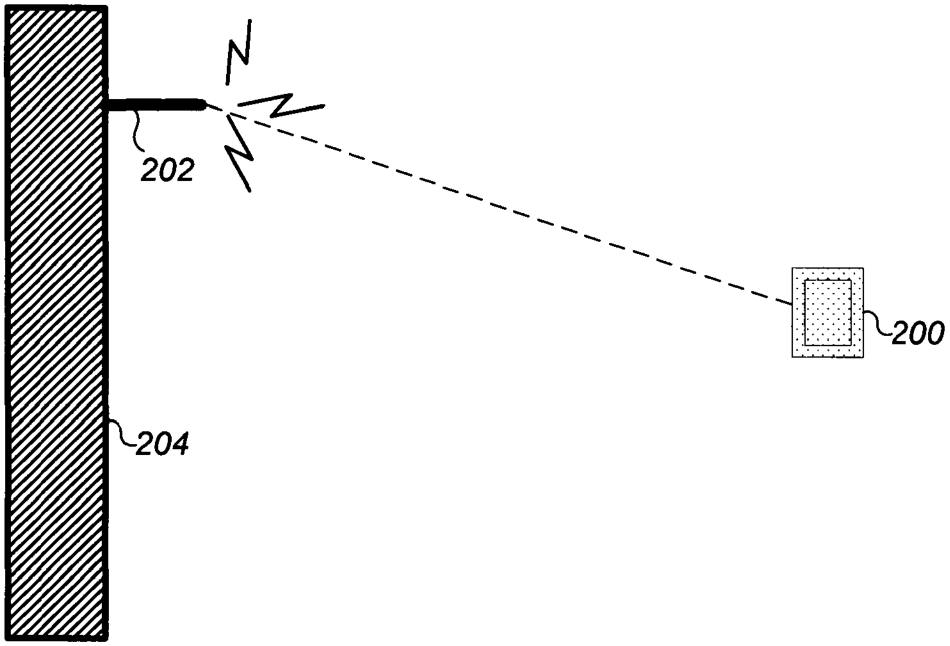


Fig. 2B

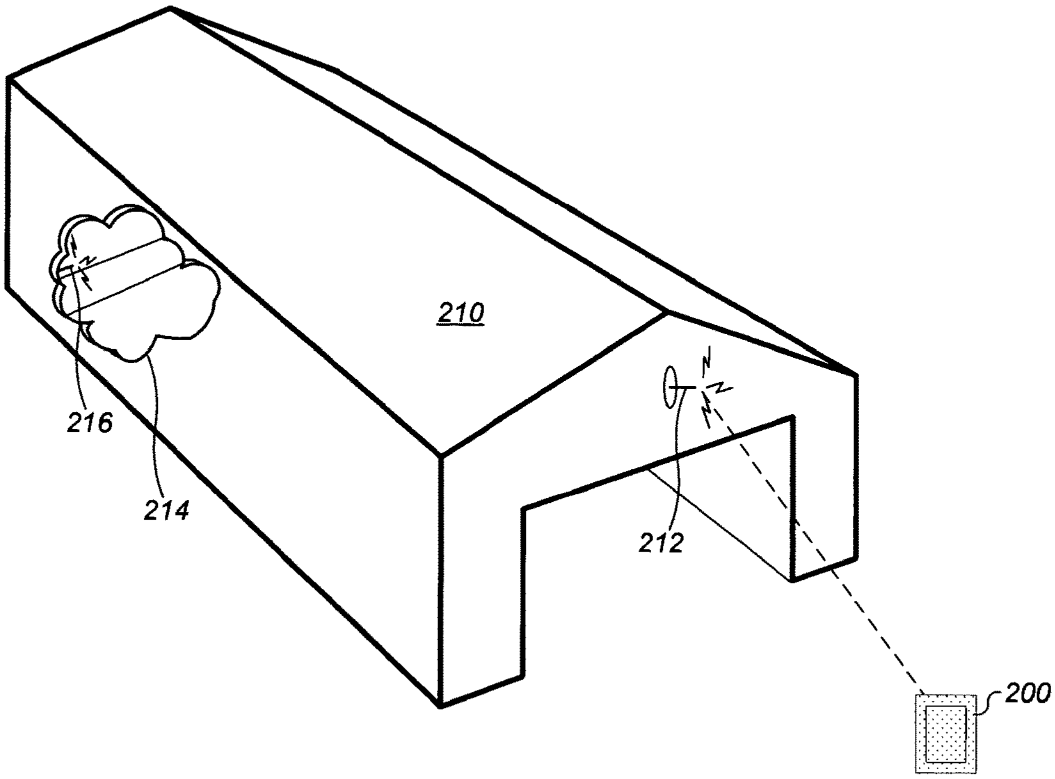


Fig. 2C

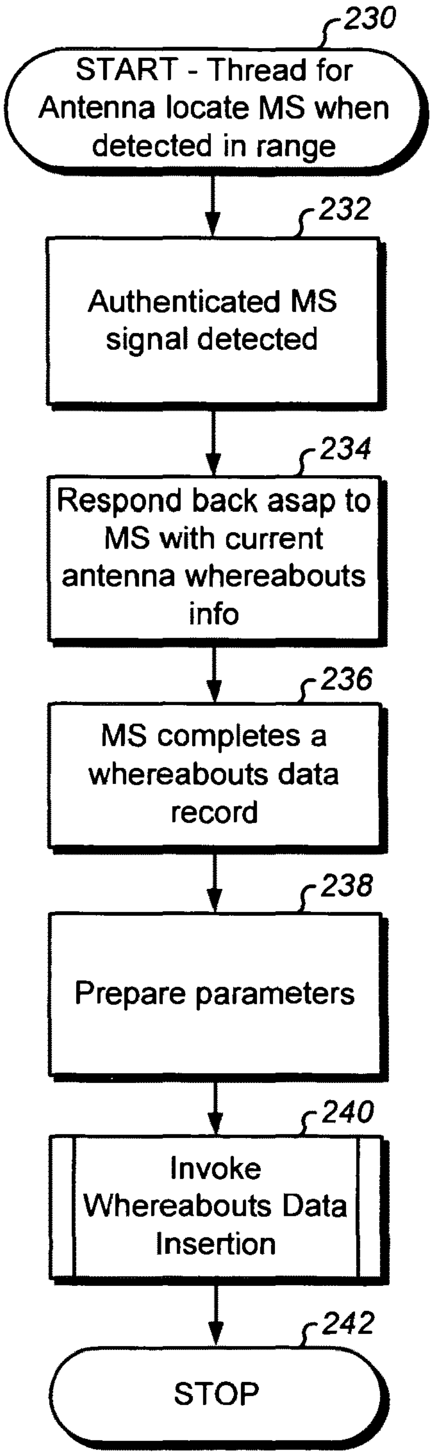

230
START - Thread for Antenna locate MS when detected in range
232
Authenticated MS signal detected
234
Respond back asap to MS with current antenna whereabouts info
236
MS completes a whereabouts data record
238
Prepare parameters
240
Invoke Whereabouts Data Insertion
242
STOP

Fig. 2D

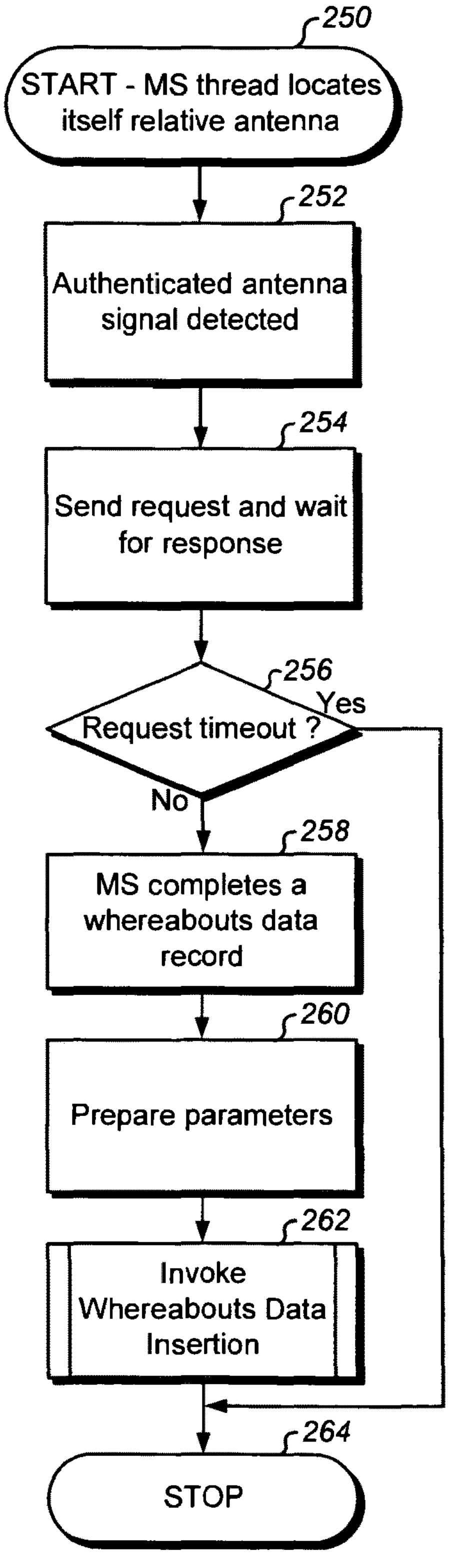

250
START - MS thread locates itself relative antenna
252
Authenticated antenna signal detected
254
Send request and wait for response
256
Request timeout ?
Yes
No
258
MS completes a whereabouts data record
260
Prepare parameters
262
Invoke Whereabouts Data Insertion
264
STOP

Fig. 2E

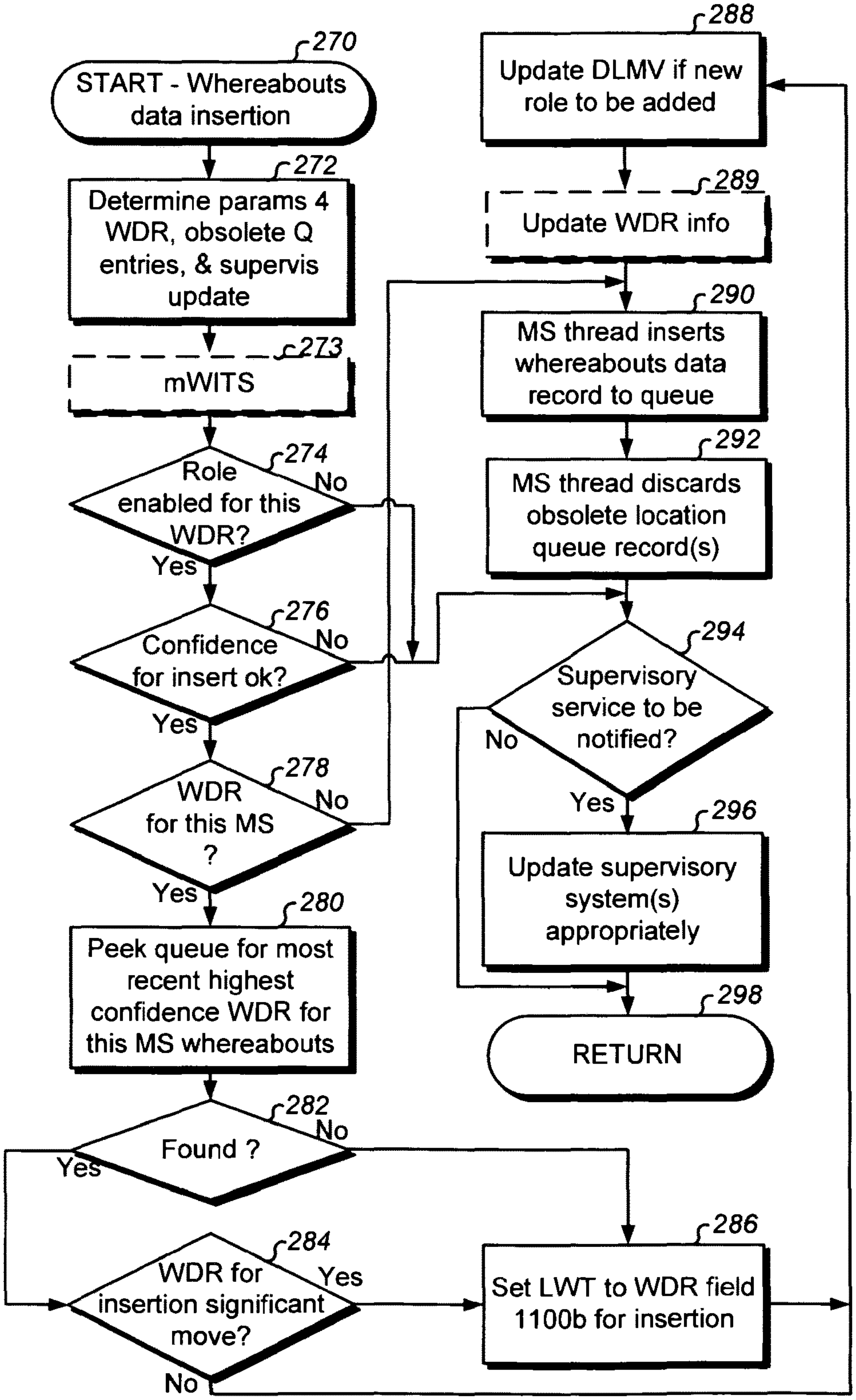

270
START - Whereabouts data insertion
272
Determine params 4 WDR, obsolete Q entries, & supervis update
273
mWITS
274
Role enabled for this WDR?
No
Yes
276
Confidence for insert ok?
No
Yes
278
WDR for this MS ?
No
Yes
280
Peek queue for most recent highest confidence WDR for this MS whereabouts
282
Found ?
No
Yes
284
WDR for insertion significant move?
Yes
No
286
Set LWT to WDR field 1100b for insertion
288
Update DLMV if new role to be added
289
Update WDR info
290
MS thread inserts whereabouts data record to queue
292
MS thread discards obsolete location queue record(s)
294
Supervisory service to be notified?
No
Yes
296
Update supervisory system(s) appropriately
298
RETURN

Fig. 2F

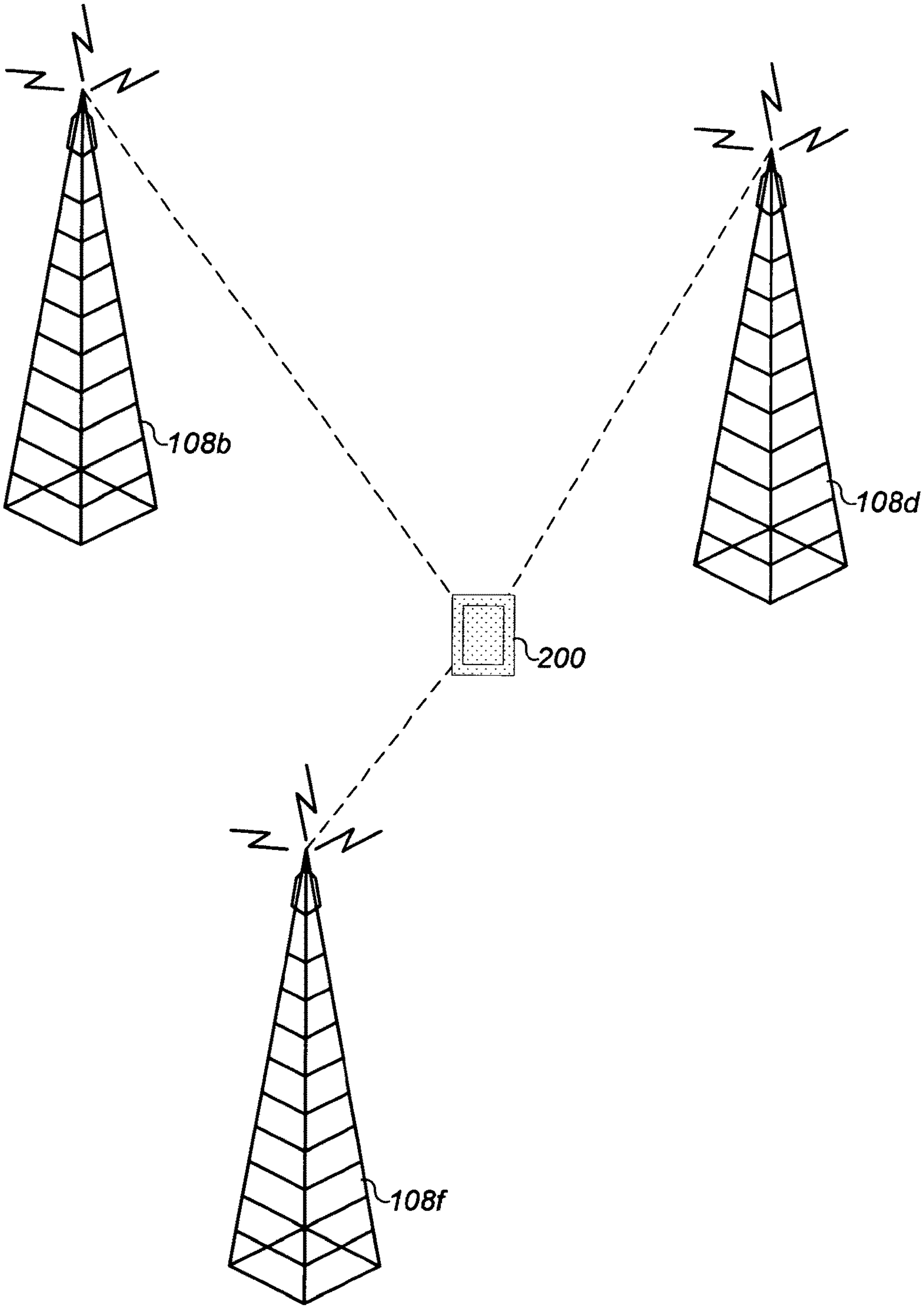


Fig. 3A

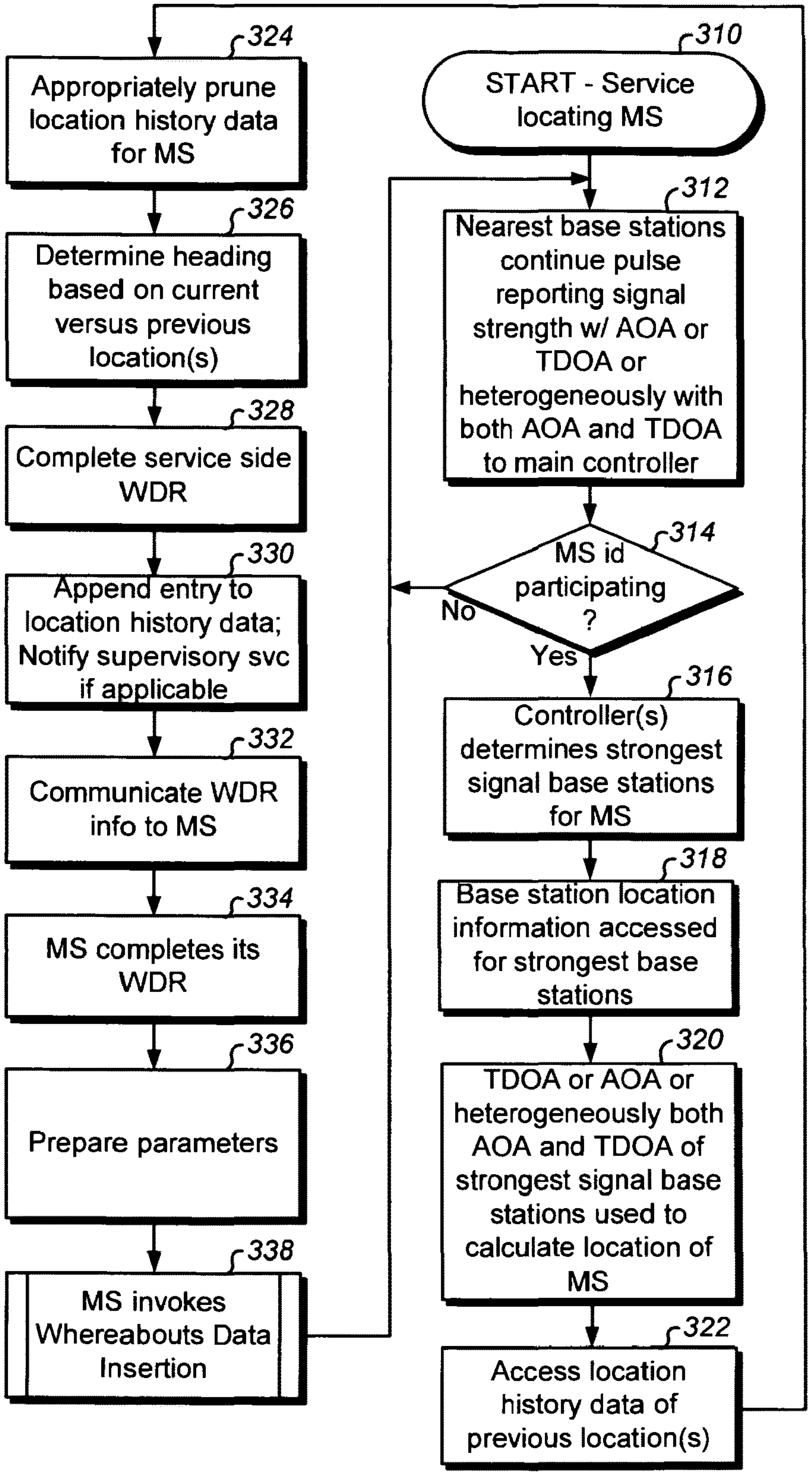

310
START - Service locating MS
312
Nearest base stations continue pulse reporting signal strength w/ AOA or TDOA or heterogeneously with both AOA and TDOA to main controller
314
MS id participating ?
No
Yes
316
Controller(s) determines strongest signal base stations for MS
318
Base station location information accessed for strongest base stations
320
TDOA or AOA or heterogeneously both AOA and TDOA of strongest signal base stations used to calculate location of MS
322
Access location history data of previous location(s)
324
Appropriately prune location history data for MS
326
Determine heading based on current versus previous location(s)
328
Complete service side WDR
330
Append entry to location history data; Notify supervisory svc if applicable
332
Communicate WDR info to MS
334
MS completes its WDR
336
Prepare parameters
338
MS invokes Whereabouts Data Insertion

Fig. 3B

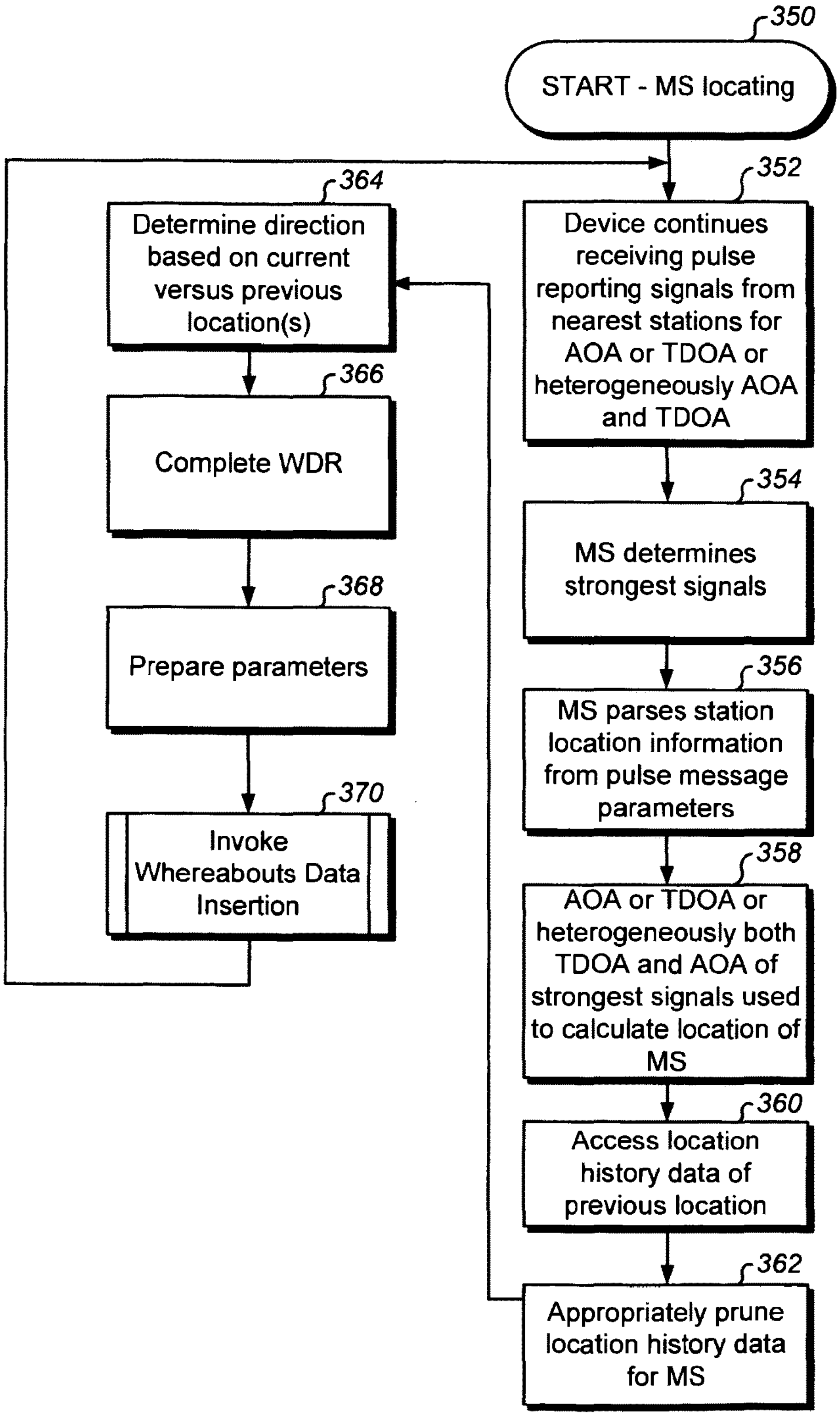

350
START - MS locating
352
Device continues receiving pulse reporting signals from nearest stations for AOA or TDOA or heterogeneously AOA and TDOA
354
MS determines strongest signals
356
MS parses station location information from pulse message parameters
358
AOA or TDOA or heterogeneously both TDOA and AOA of strongest signals used to calculate location of MS
360
Access location history data of previous location
362
Appropriately prune location history data for MS
364
Determine direction based on current versus previous location(s)
366
Complete WDR
368
Prepare parameters
370
Invoke Whereabouts Data Insertion

Fig. 3C

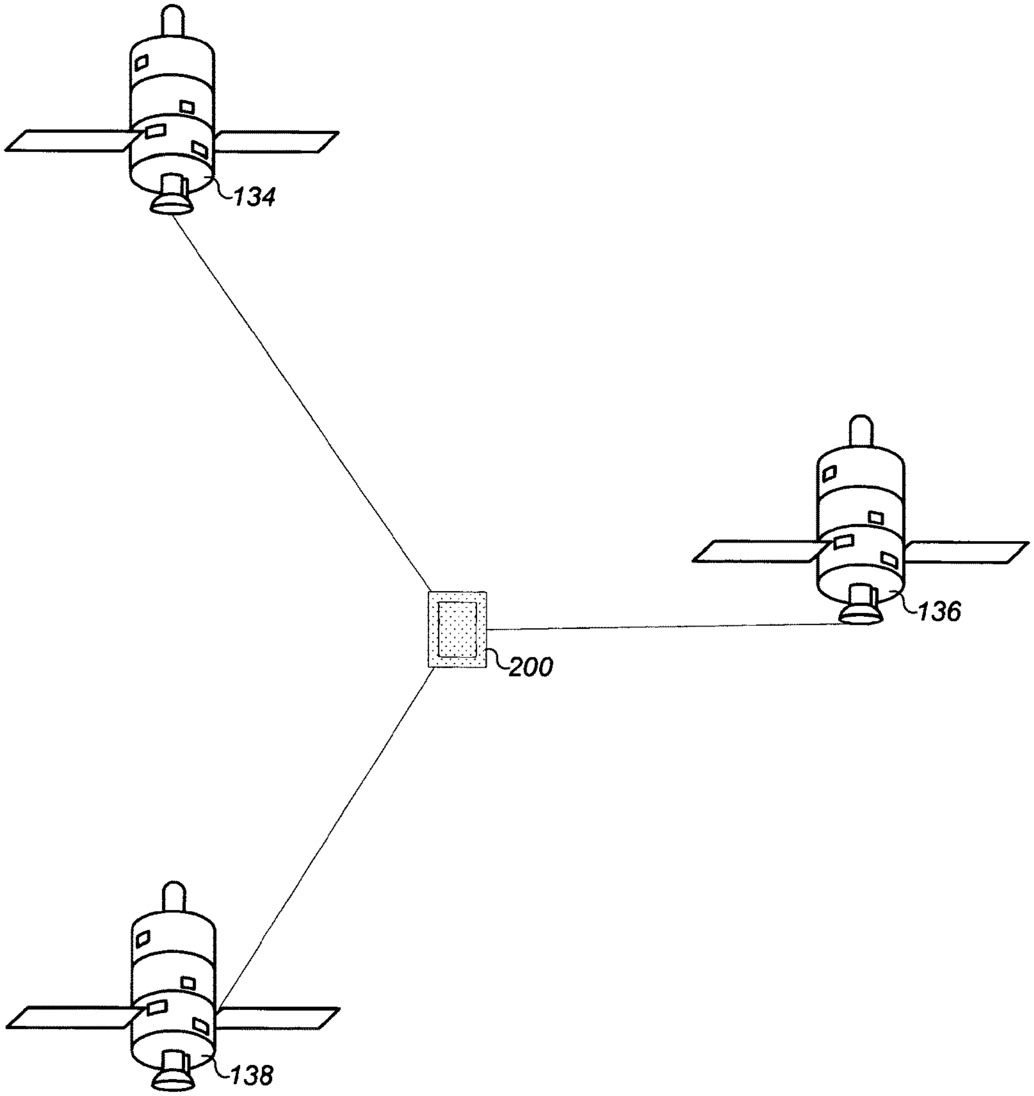


Fig. 4A

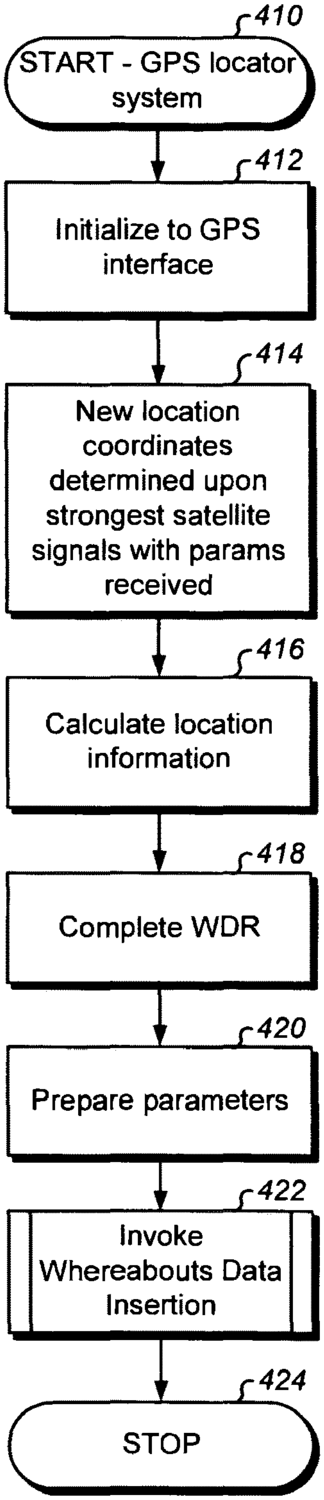

410
START - GPS locator system
412
Initialize to GPS interface
414
New location coordinates determined upon strongest satellite signals with params received
416
Calculate location information
418
Complete WDR
420
Prepare parameters
422
Invoke Whereabouts Data Insertion
424
STOP

Fig. 4B

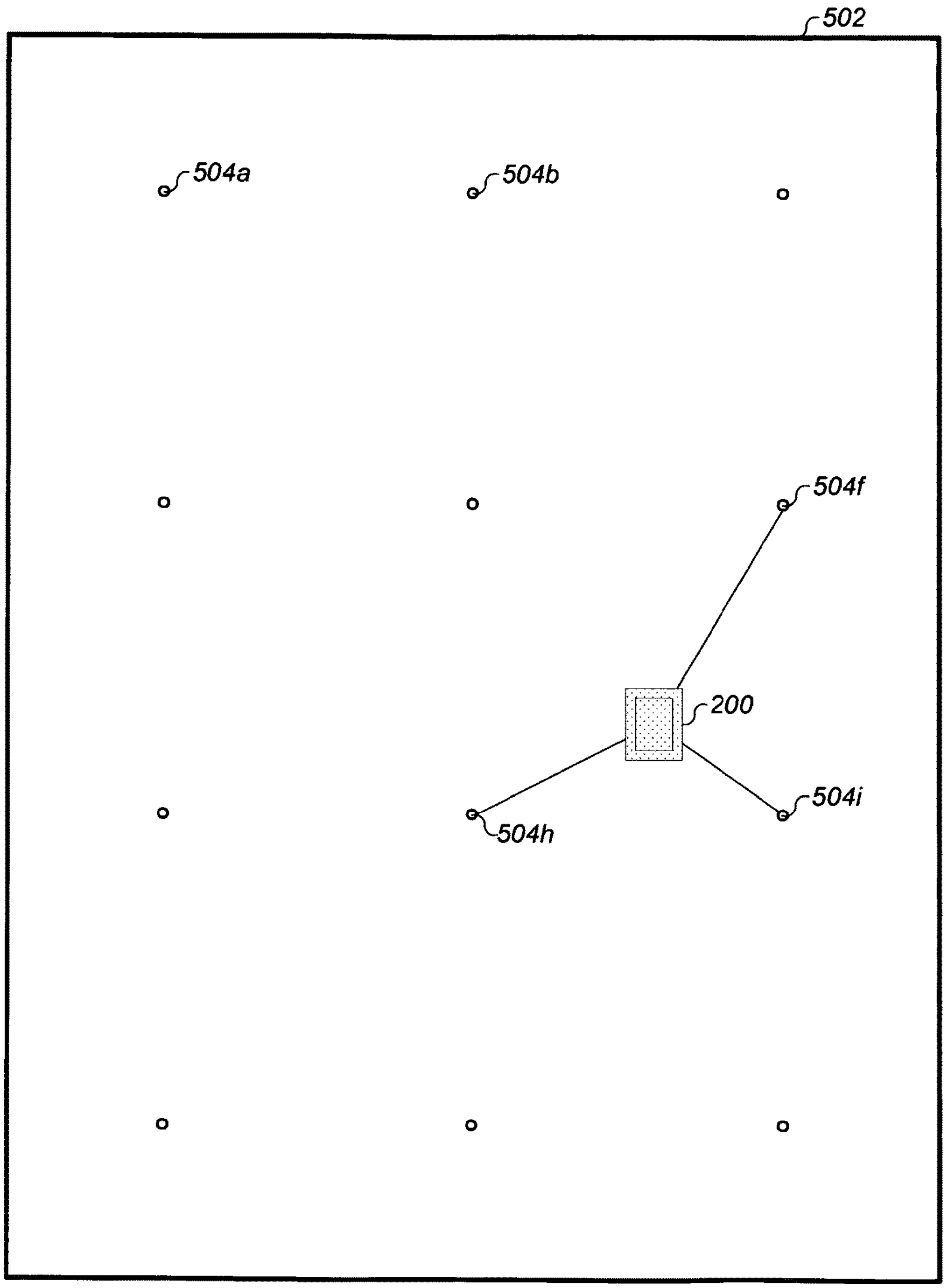


Fig. 5A

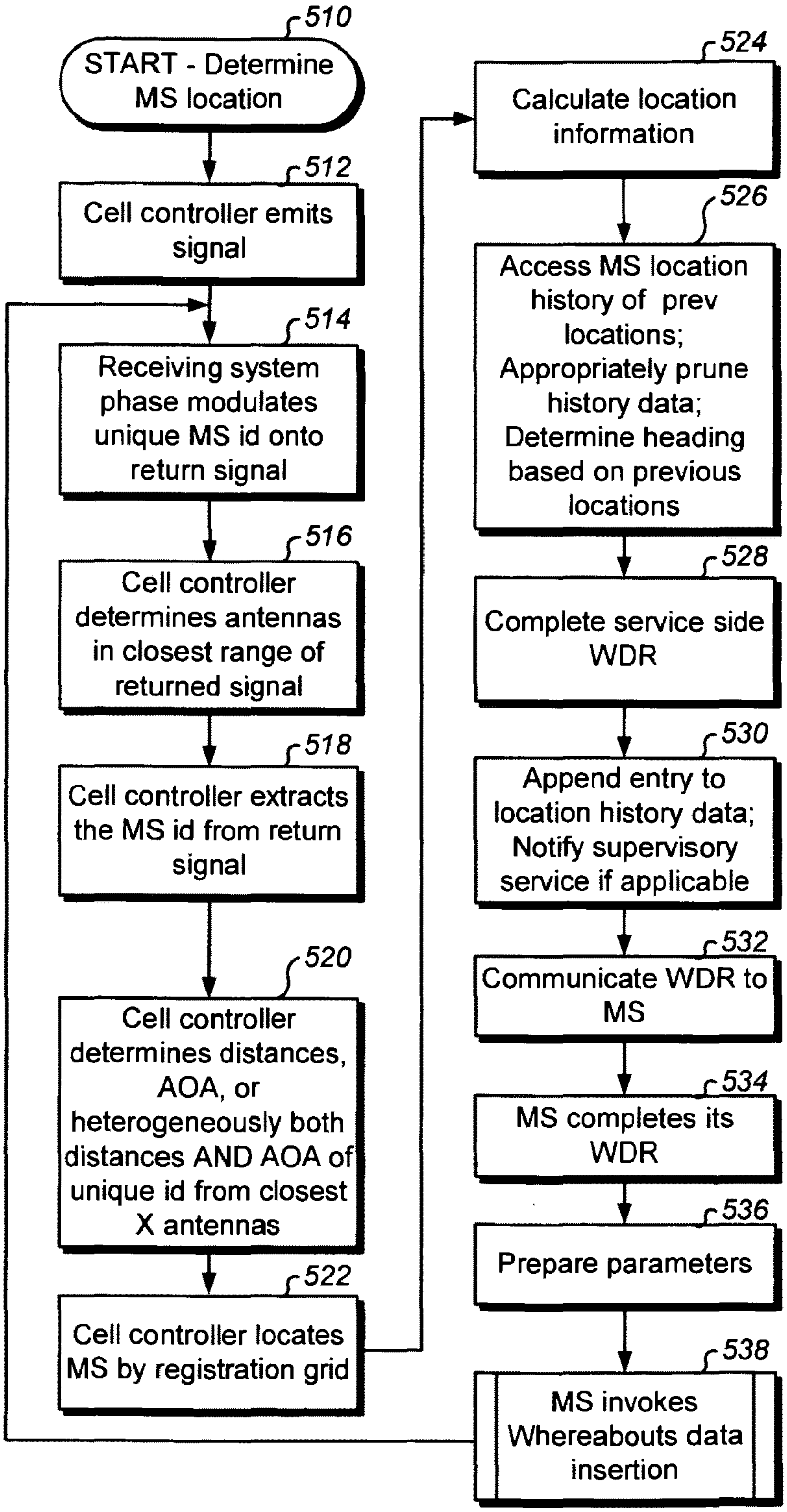

510
START - Determine MS location
512
Cell controller emits signal
514
Receiving system phase modulates unique MS id onto return signal
516
Cell controller determines antennas in closest range of returned signal
518
Cell controller extracts the MS id from return signal
520
Cell controller determines distances, AOA, or heterogeneously both distances AND AOA of unique id from closest X antennas
522
Cell controller locates MS by registration grid
524
Calculate location information
526
Access MS location history of prev locations; Appropriately prune history data; Determine heading based on previous locations
528
Complete service side WDR
530
Append entry to location history data; Notify supervisory service if applicable
532
Communicate WDR to MS
534
MS completes its WDR
536
Prepare parameters
538
MS invokes Whereabouts data insertion

Fig. 5B

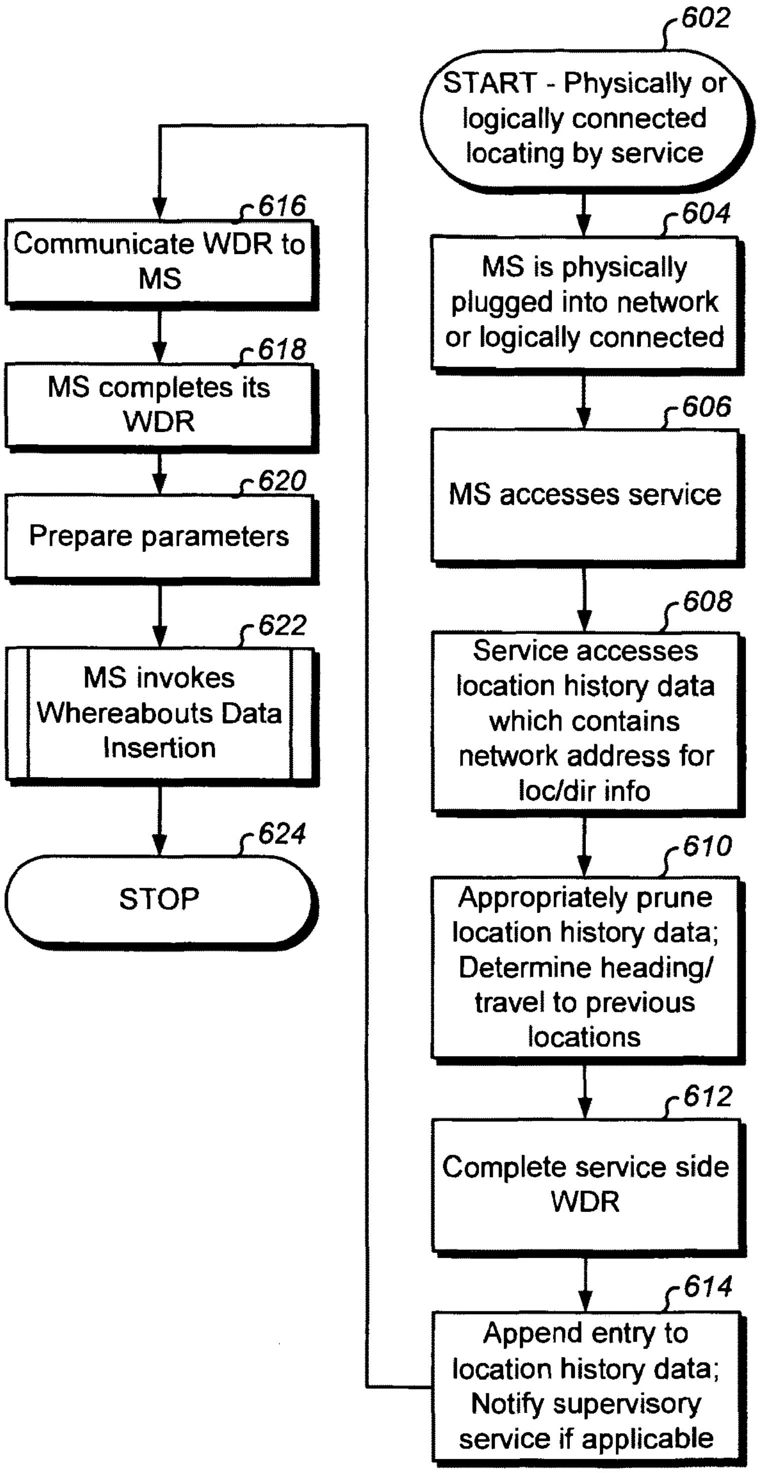

602
START - Physically or logically connected locating by service
604
MS is physically plugged into network or logically connected
606
MS accesses service
608
Service accesses location history data which contains network address for loc/dir info
610
Appropriately prune location history data; Determine heading/ travel to previous locations
612
Complete service side WDR
614
Append entry to location history data; Notify supervisory service if applicable
616
Communicate WDR to MS
618
MS completes its WDR
620
Prepare parameters
622
MS invokes Whereabouts Data Insertion
624
STOP

Fig. 6A

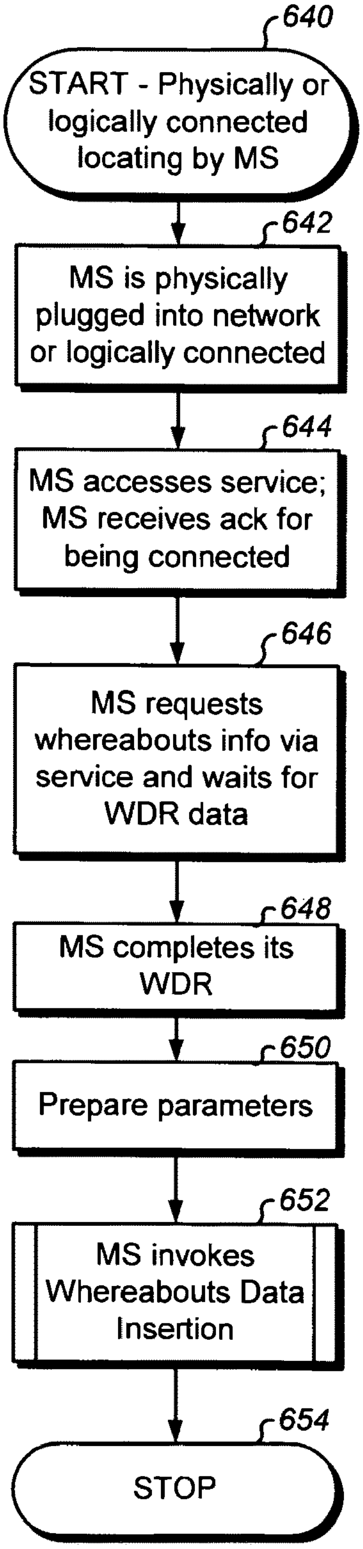

640
START - Physically or logically connected locating by MS
642
MS is physically plugged into network or logically connected
644
MS accesses service; MS receives ack for being connected
646
MS requests whereabouts info via service and waits for WDR data
648
MS completes its WDR
650
Prepare parameters
652
MS invokes Whereabouts Data Insertion
654
STOP

Fig. 6B

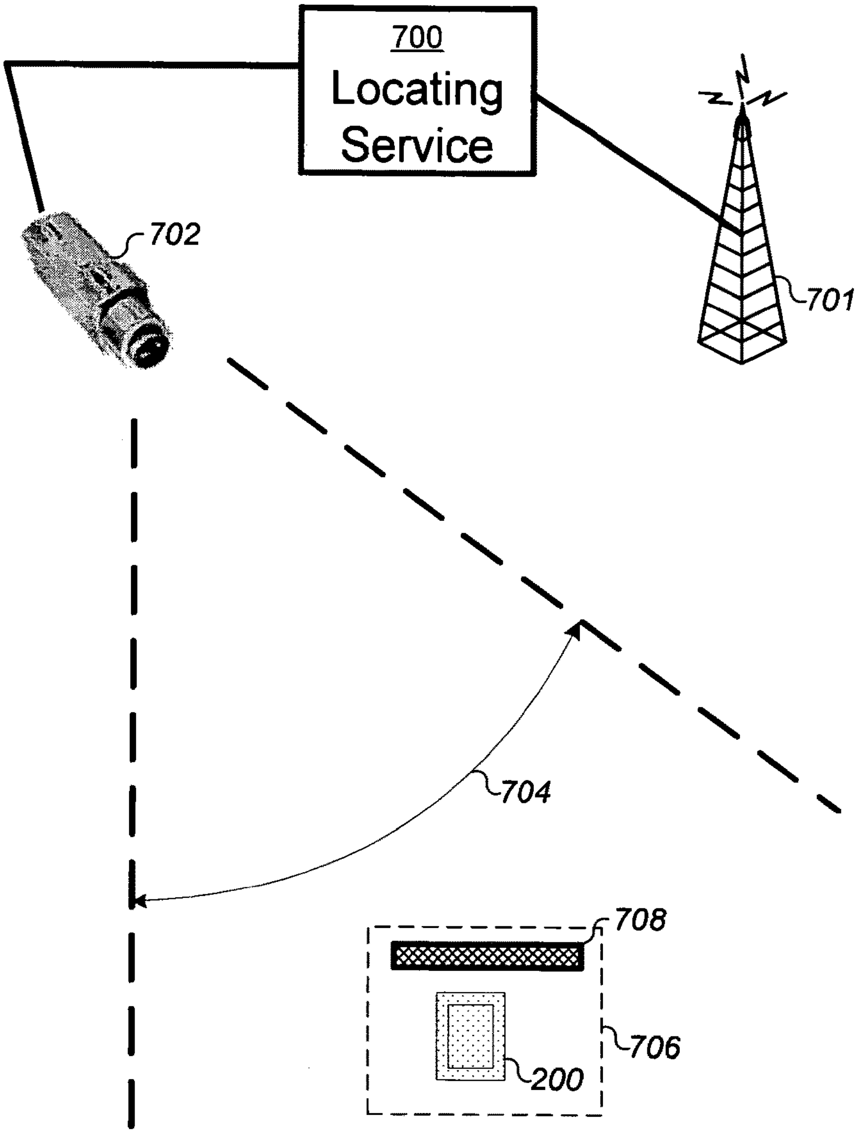


Fig. 7A

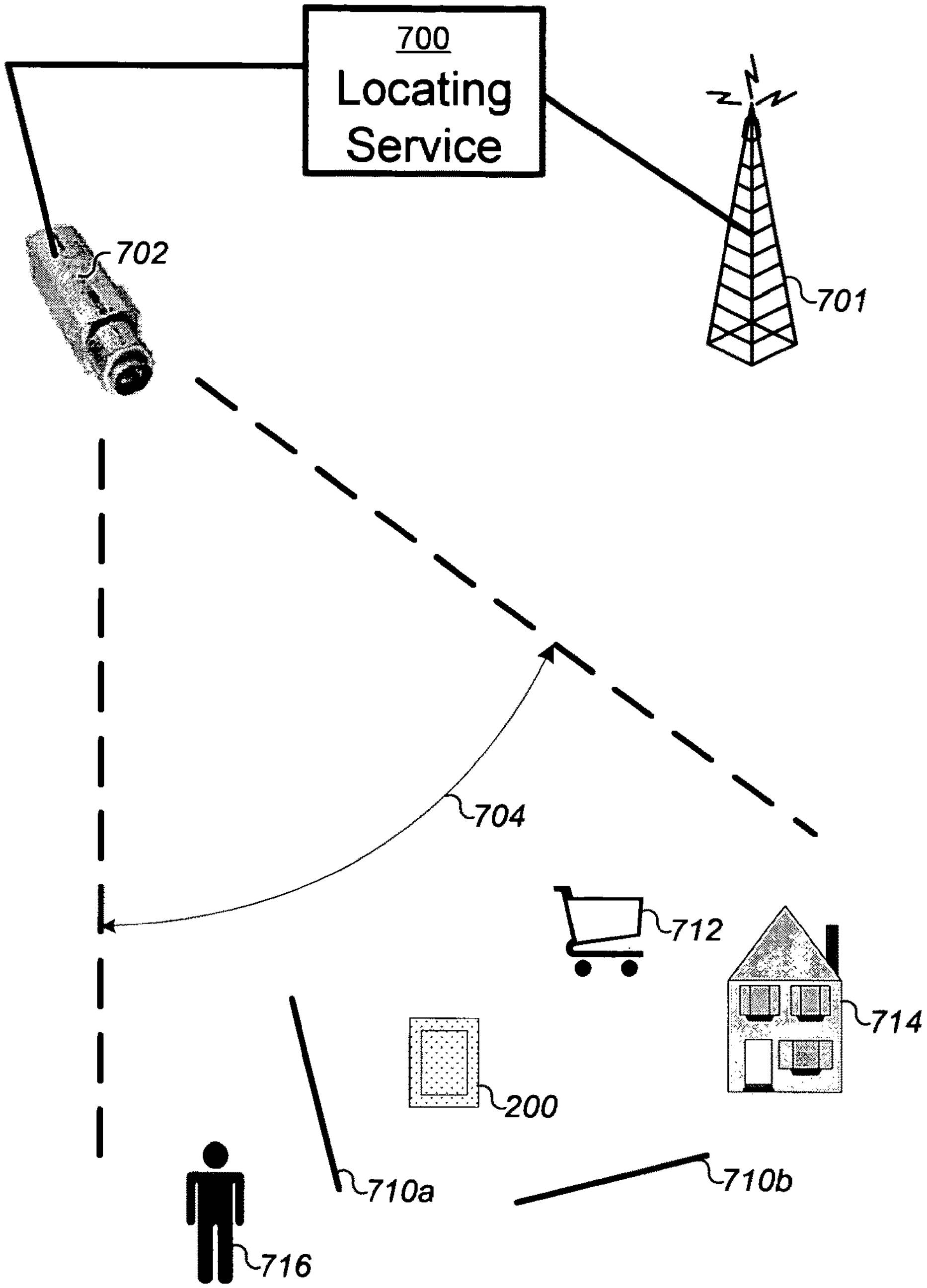


Fig. 7B

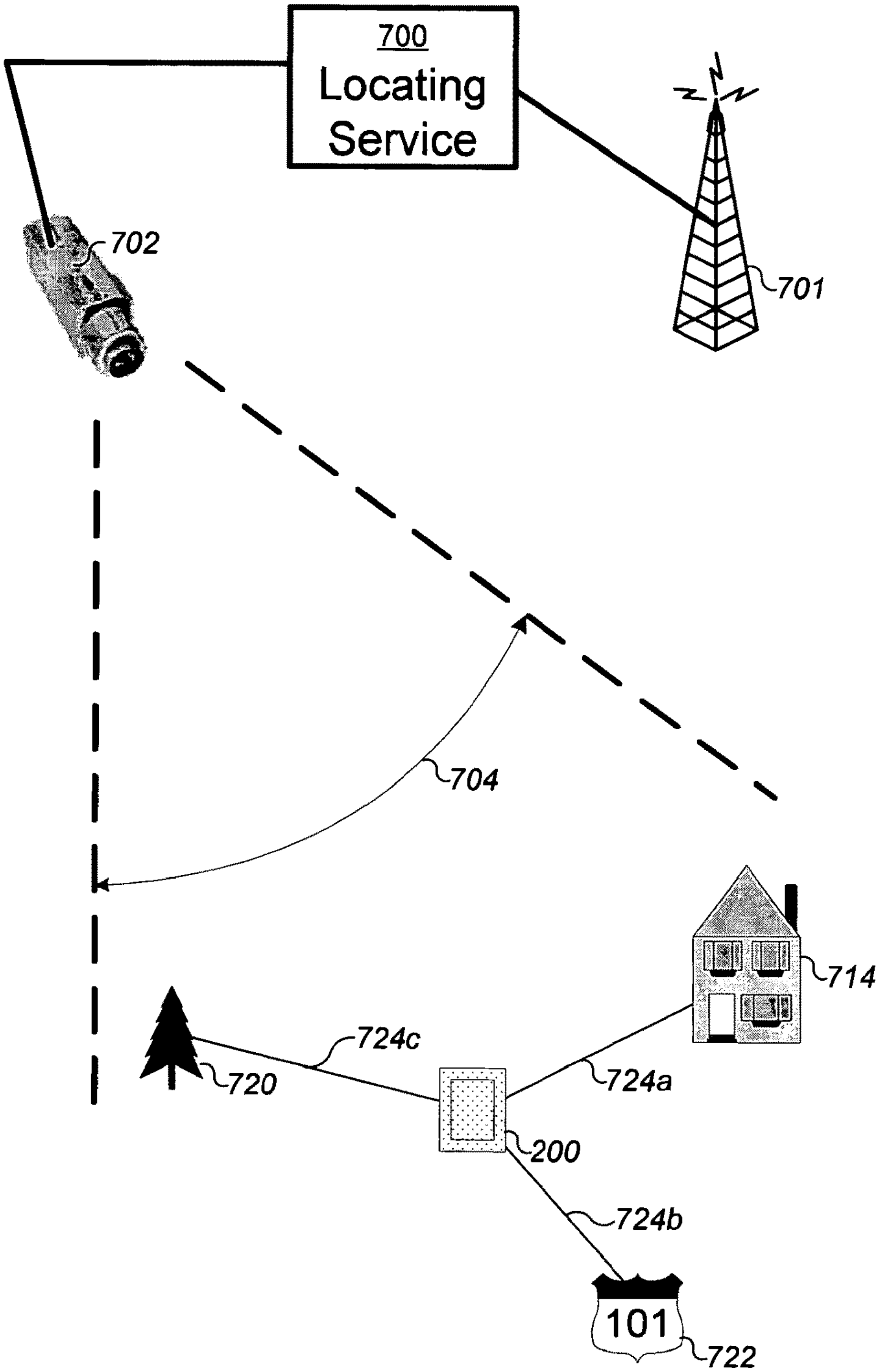

700
Locating Service
702
701
704
714
724c
720
724a
200
724b
101
722

Fig. 7C

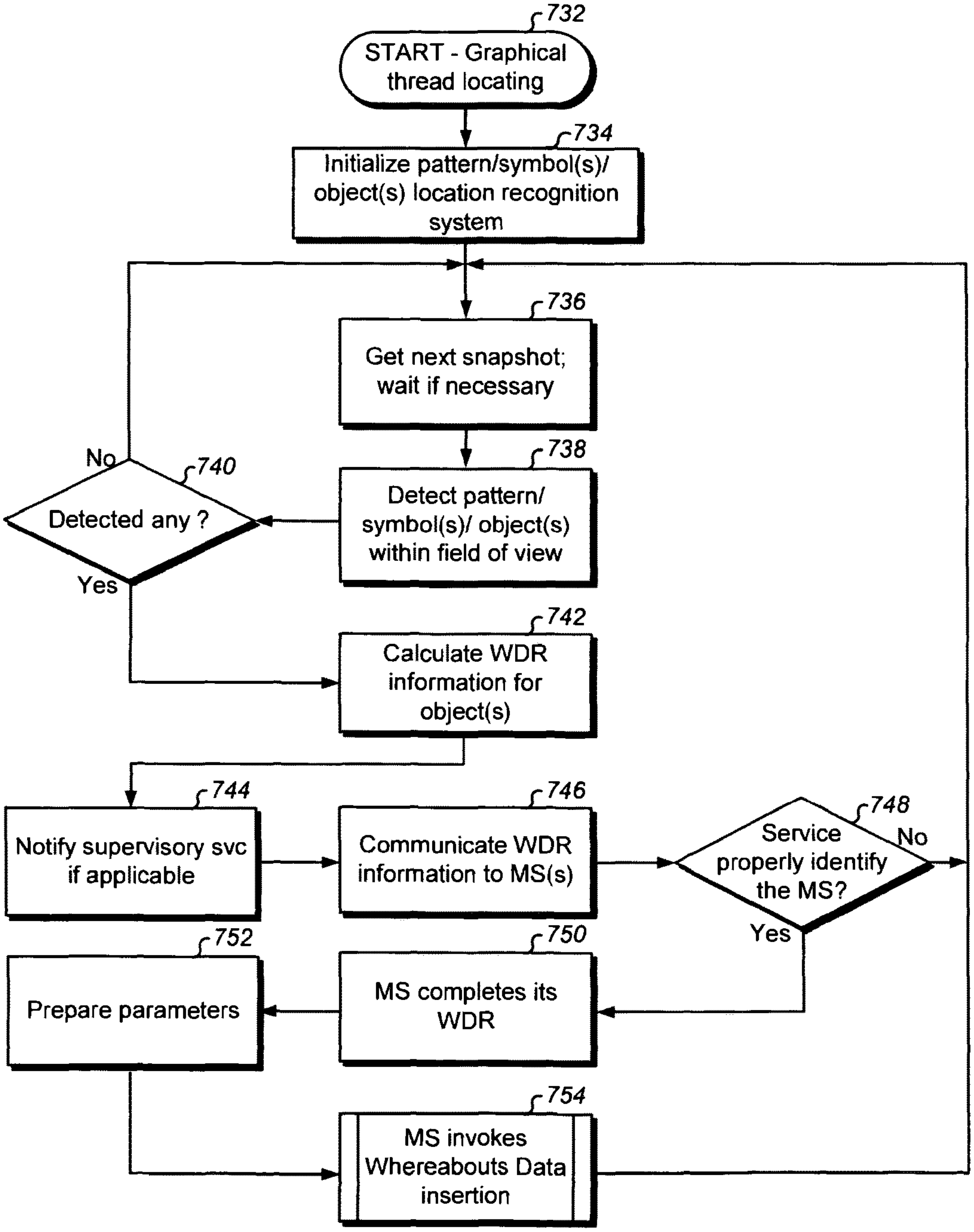

732
START - Graphical thread locating
734
Initialize pattern/symbol(s)/ object(s) location recognition system
736
Get next snapshot; wait if necessary
738
Detect pattern/ symbol(s)/ object(s) within field of view
740
Detected any ?
No
Yes
742
Calculate WDR information for object(s)
744
Notify supervisory svc if applicable
746
Communicate WDR information to MS(s)
748
Service properly identify the MS?
No
Yes
750
MS completes its WDR
752
Prepare parameters
754
MS invokes Whereabouts Data insertion

Fig. 7D

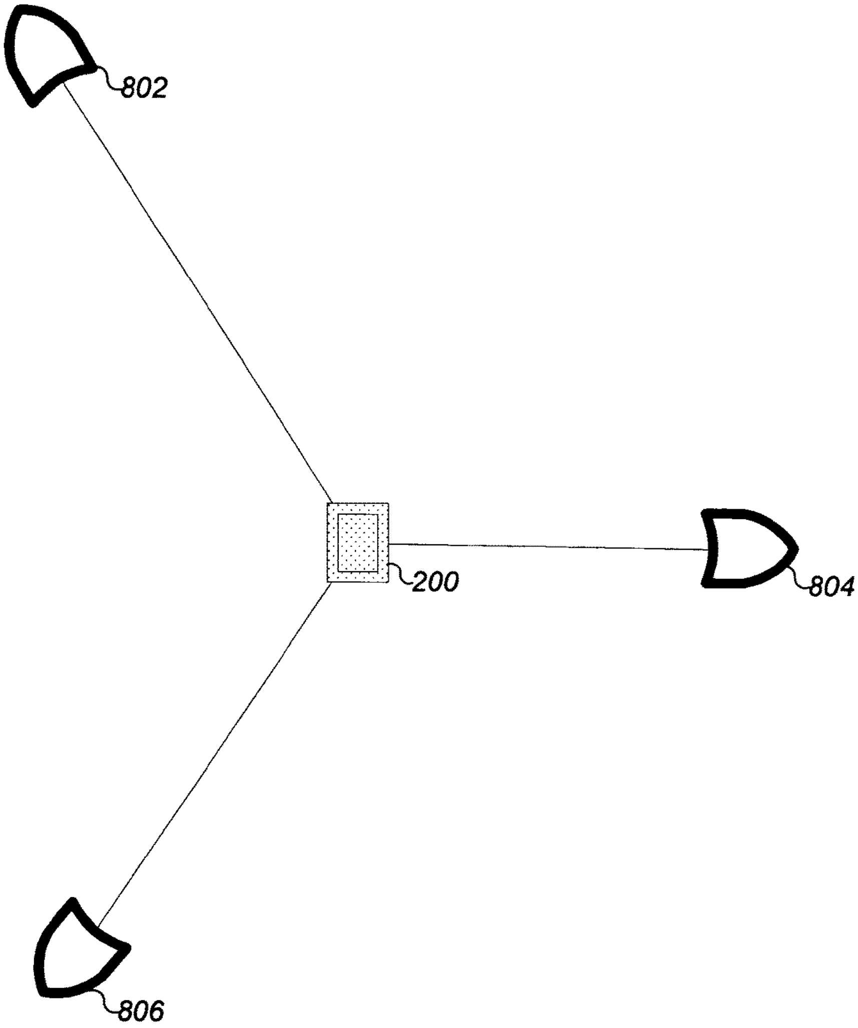


Fig. 8A

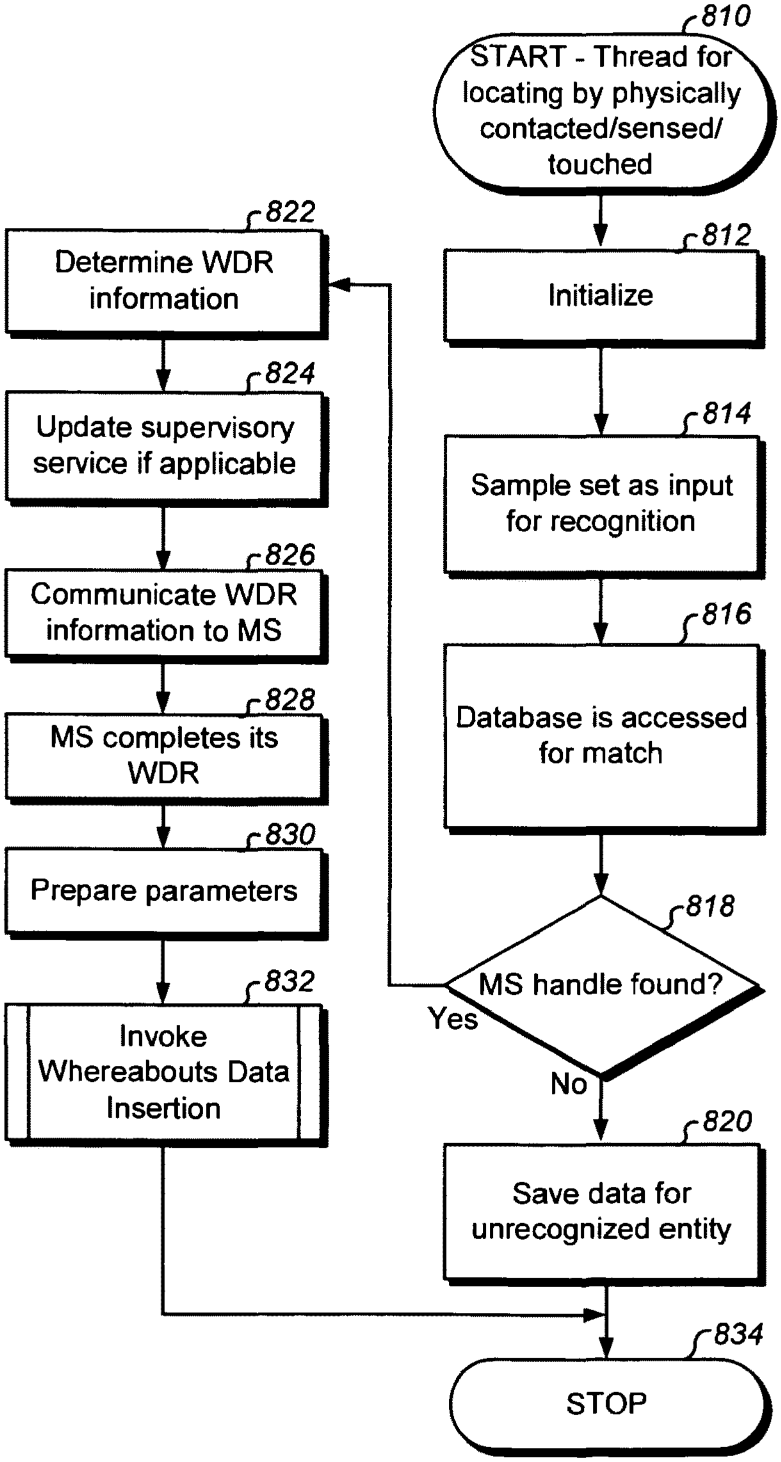

810
START - Thread for locating by physically contacted/sensed/ touched
812
Initialize
814
Sample set as input for recognition
816
Database is accessed for match
818
MS handle found?
Yes
No
820
Save data for unrecognized entity
822
Determine WDR information
824
Update supervisory service if applicable
826
Communicate WDR information to MS
828
MS completes its WDR
830
Prepare parameters
832
Invoke Whereabouts Data Insertion
834
STOP

Fig. 8B

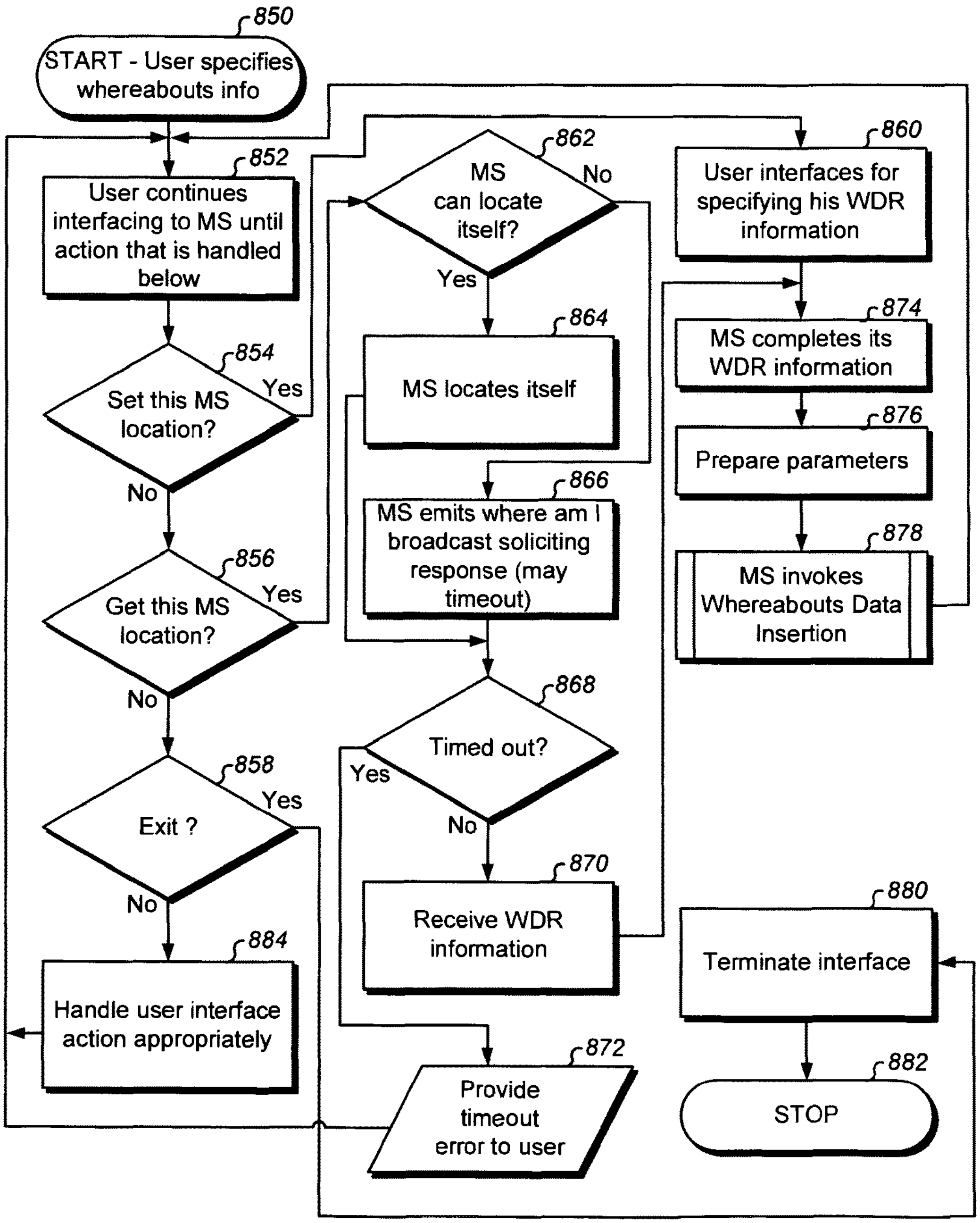

850
START - User specifies whereabouts info
852
User continues interfacing to MS until action that is handled below
854
Set this MS location?
Yes
No
856
Get this MS location?
Yes
No
858
Exit ?
Yes
No
884
Handle user interface action appropriately
862
MS can locate itself?
No
Yes
864
MS locates itself
866
MS emits where am I broadcast soliciting response (may timeout)
868
Timed out?
Yes
No
870
Receive WDR information
872
Provide timeout error to user
860
User interfaces for specifying his WDR information
874
MS completes its WDR information
876
Prepare parameters
878
MS invokes Whereabouts Data Insertion
880
Terminate interface
882
STOP

Fig. 8C

		MS (id 0A12:43EF:985B:012F)
GPS	C	x
	S	
A-GPS	C	
	S	
D-GPS	C	
	S	
Graphic-Pattern(s)	C	
	S	
Graphic-Distances	C	
	S	
Graphic-Triangulate	C	
	S	
Artificial Intelligence	C	
	S	
Cell Range	C	
	S	x
Cell AOA	C	
	S	
Cell TDOA	C	
	S	x
Cell MPT	C	
	S	x
Antenna Range	C	
	S	x
Antenna AOA	C	
	S	x
Antenna TDOA	C	
	S	x
Antenna MPT	C	
	S	x
LIDAR/optics	C	
	S	
Manual	C	
	S	
Contact	C	
	S	x
MPT	C	
	S	x
Client Logical Connect	C	
	S	
Server Logical Connect	C	
	S	
Client Physical Connect	C	
	S	
Server Physical Connect	C	
	S	
Sound/Acoustics	C	
	S	
Microdot/ RFI	C	
	S	
Transponder	C	
	S	
Others	C	
	S	
. . .	C	
	S	

Fig. 9A

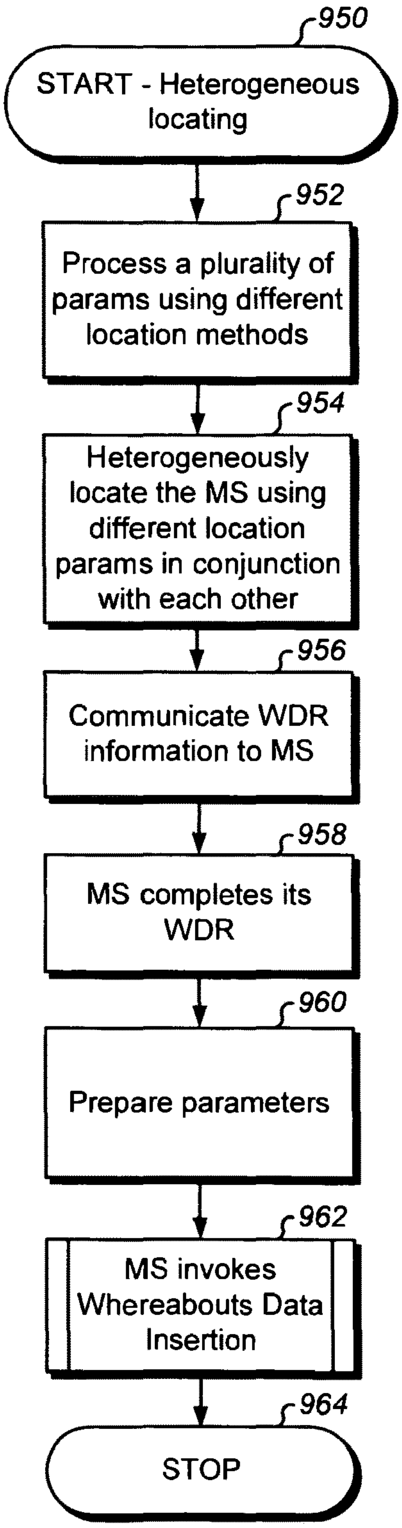

950
START - Heterogeneous locating
952
Process a plurality of params using different location methods
954
Heterogeneously locate the MS using different location params in conjunction with each other
956
Communicate WDR information to MS
958
MS completes its WDR
960
Prepare parameters
962
MS invokes Whereabouts Data Insertion
964
STOP

Fig. 9B

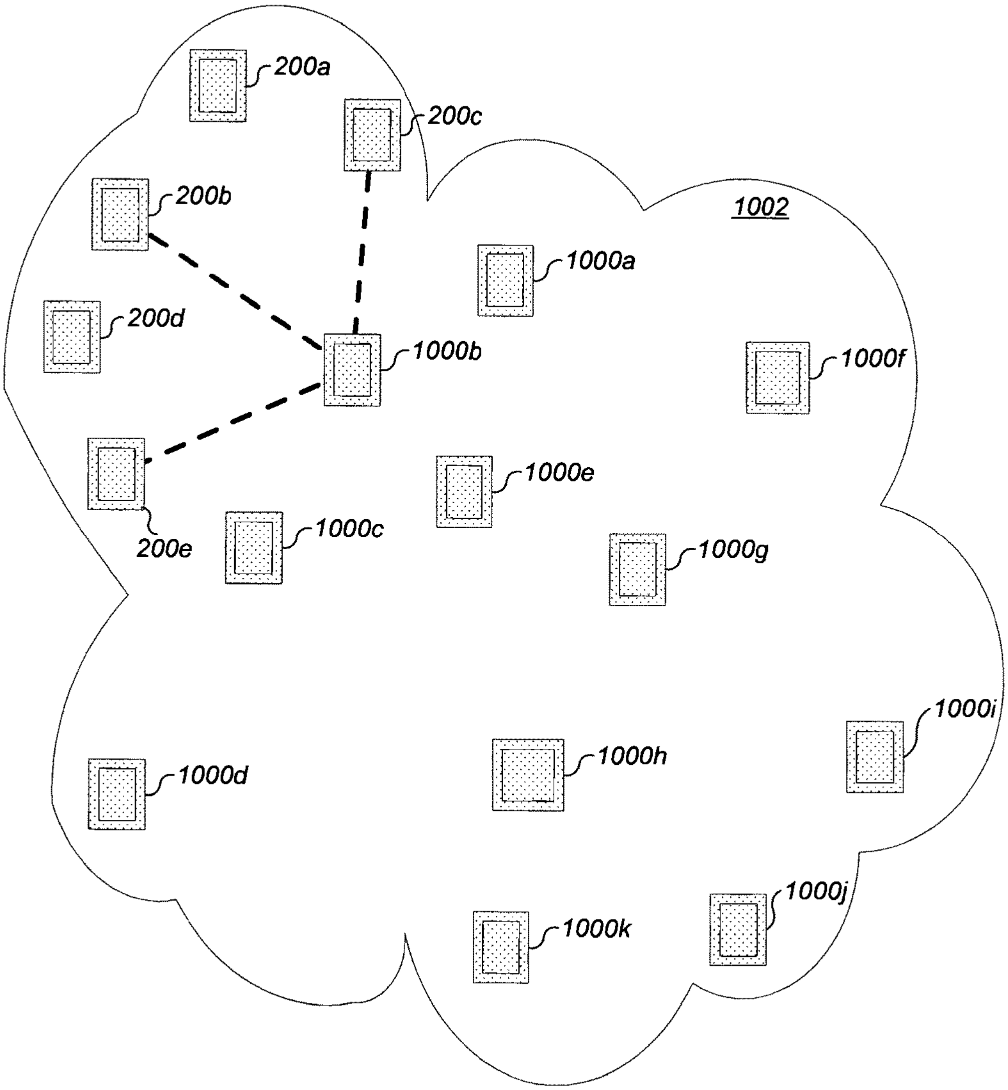

200a
200c
200b
1002
1000a
200d
1000b
1000f
1000e
200e
1000c
1000g
1000i
1000h
1000d
1000j
1000k

Fig. 10A

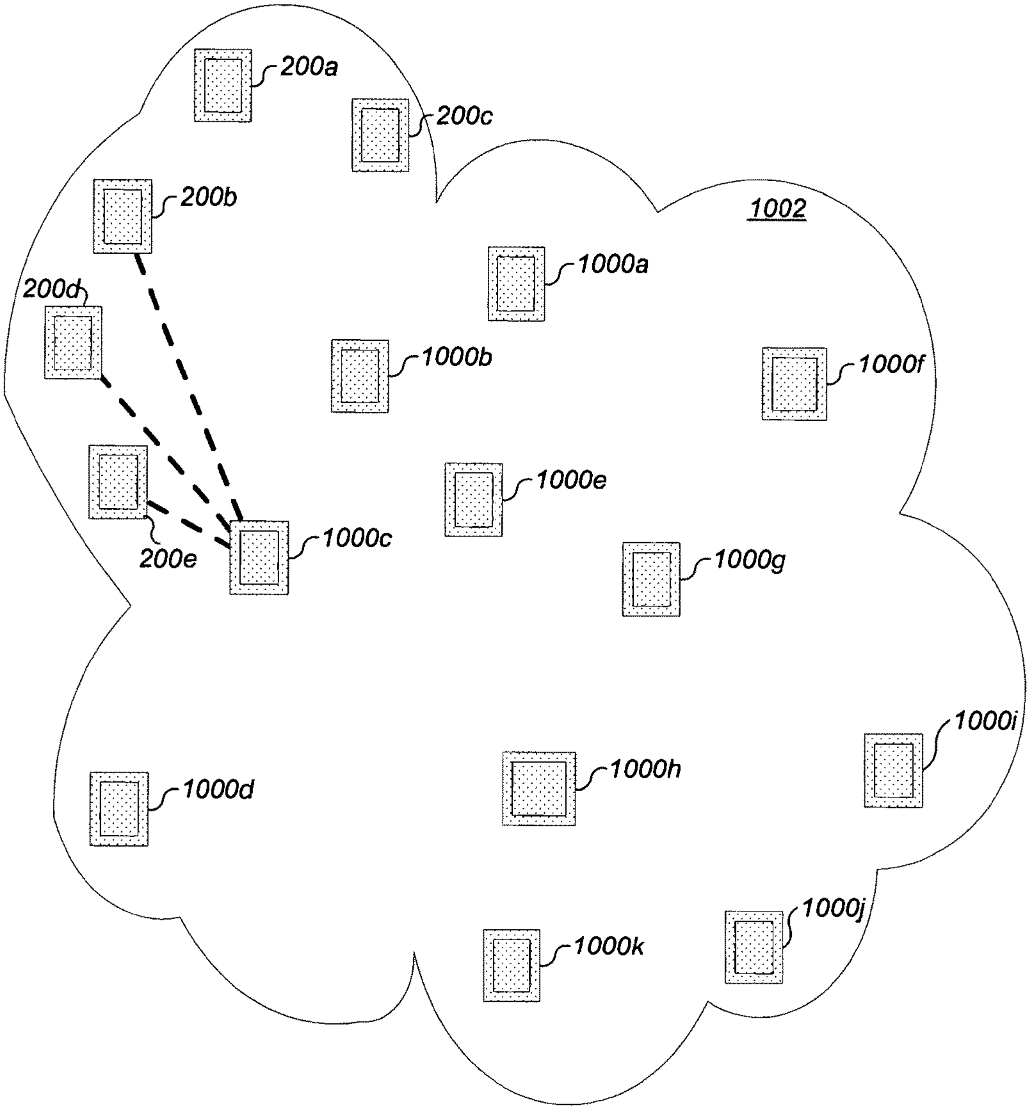

200a
200c
200b
1002
1000a
200d
1000b
1000f
1000e
1000c
200e
1000g
1000i
1000h
1000d
1000j
1000k

Fig. 10B

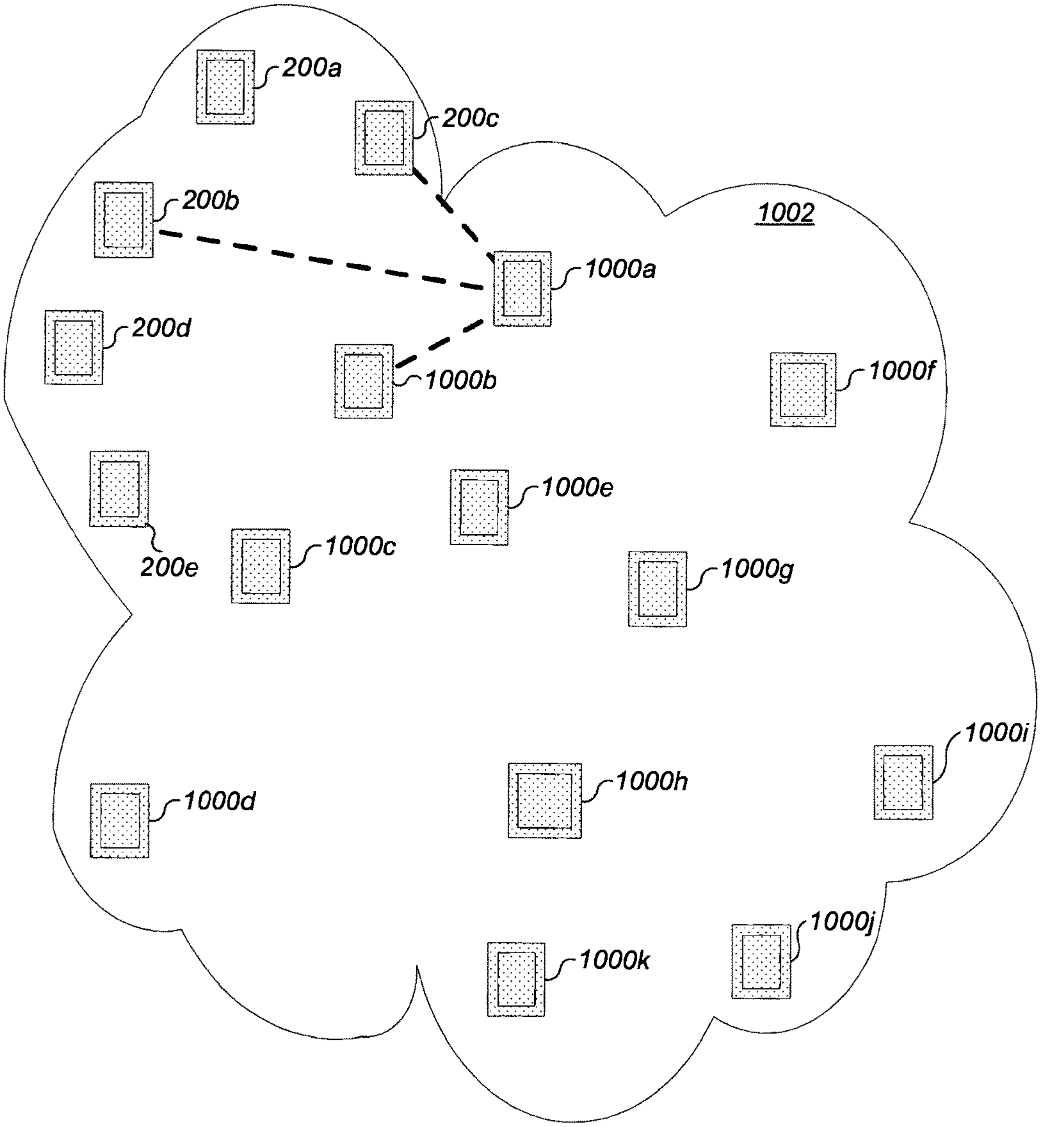

200a
200c
200b
1002
1000a
200d
1000b
1000f
1000e
200e
1000c
1000g
1000i
1000h
1000d
1000j
1000k

Fig. 10C

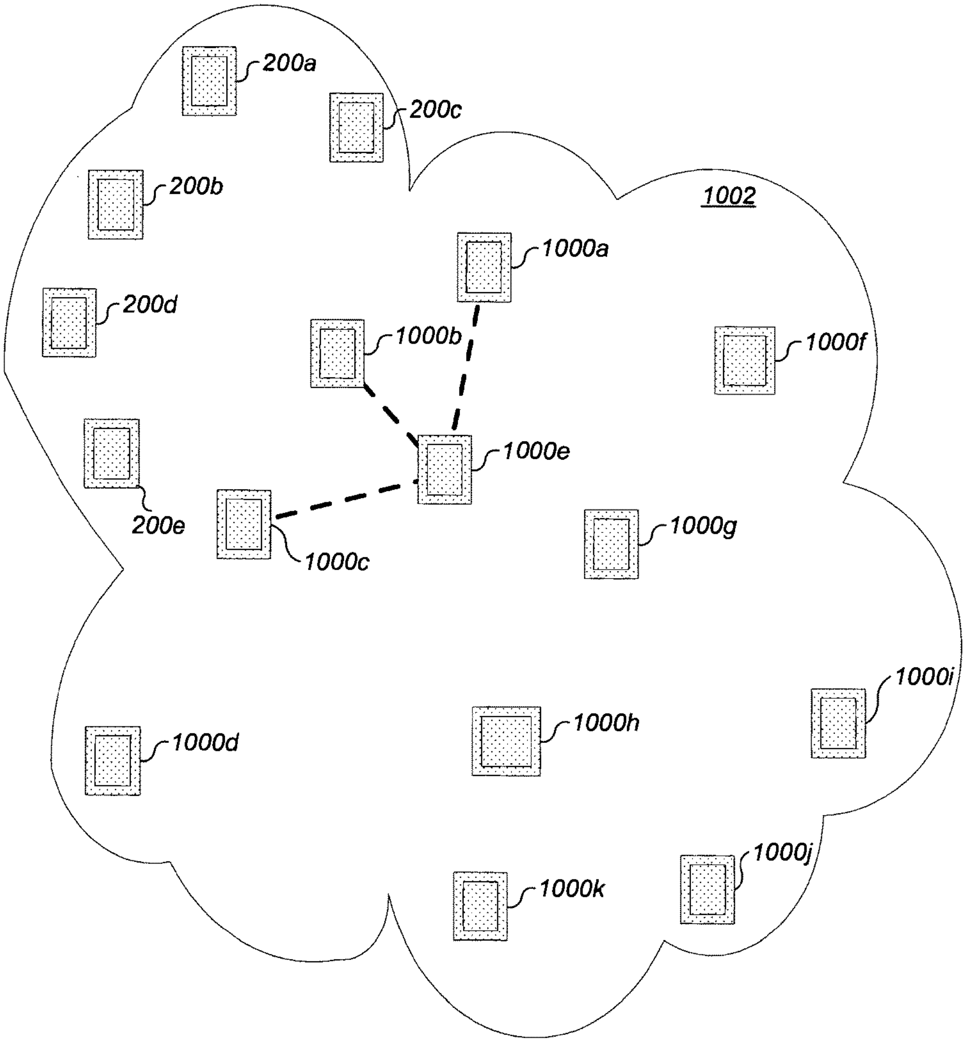

200a
200c
200b
1002
1000a
200d
1000b
1000f
1000e
200e
1000g
1000c
1000i
1000h
1000d
1000j
1000k

Fig. 10D

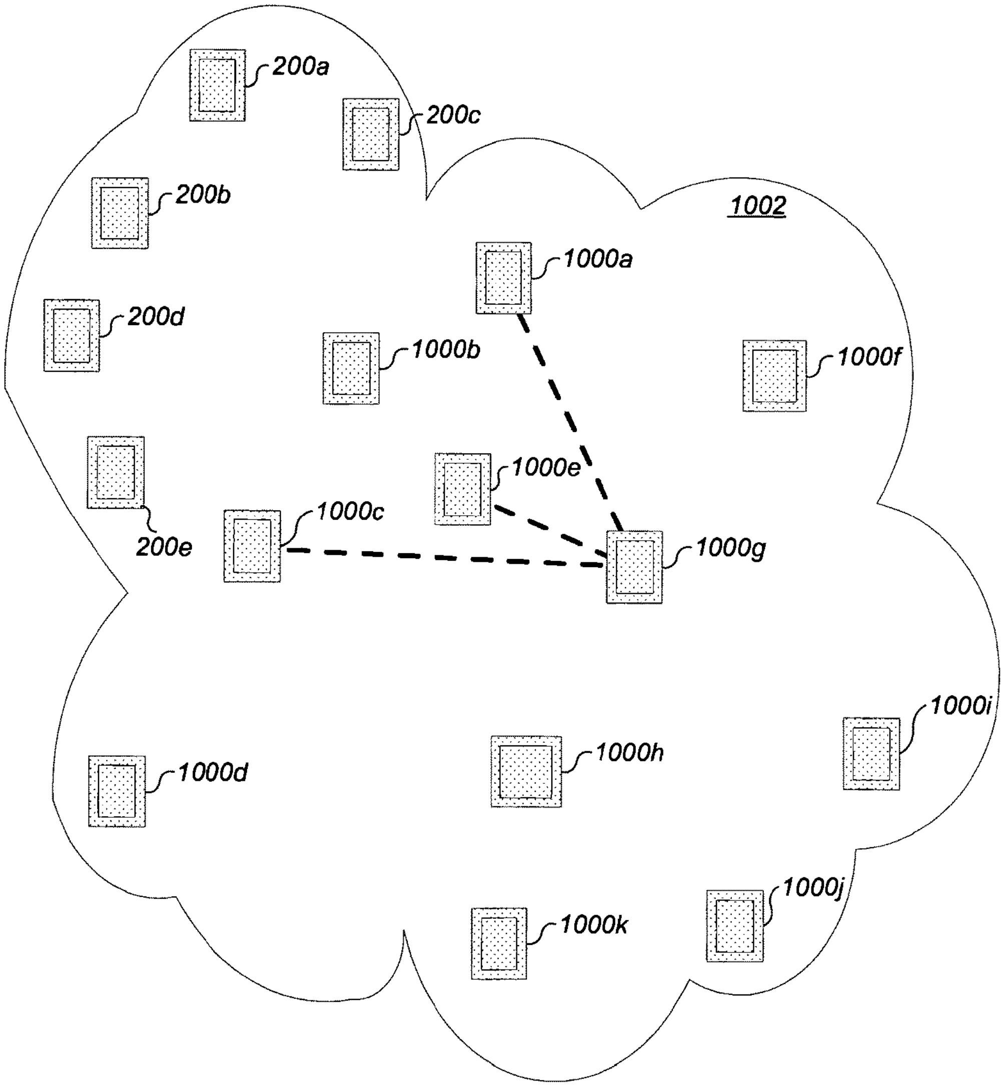

200a
200c
200b
200d
200e
1002
1000a
1000b
1000f
1000e
1000c
1000g
1000i
1000d
1000h
1000j
1000k

Fig. 10E

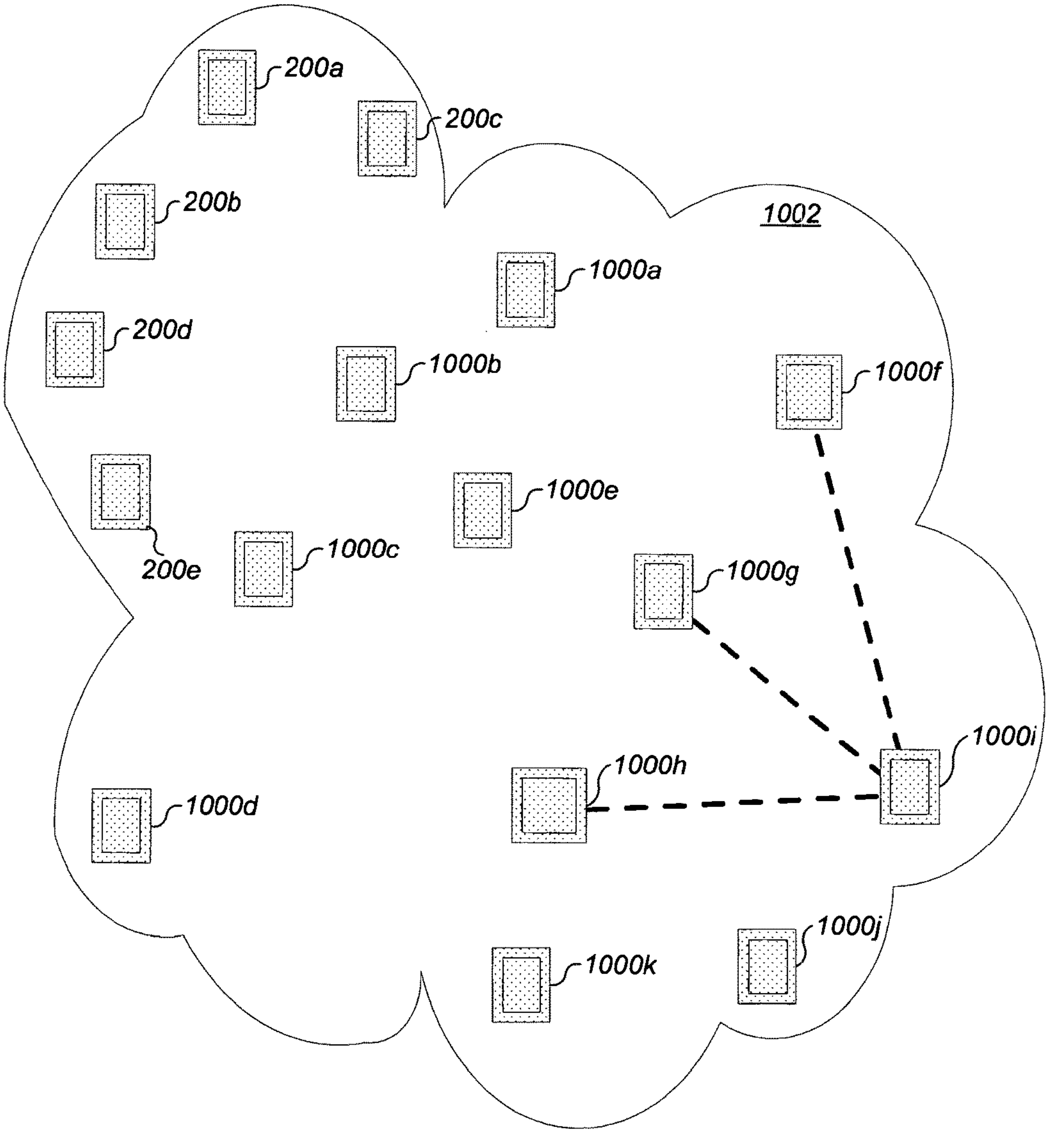

200a
200c
200b
1002
1000a
200d
1000b
1000f
1000e
200e
1000c
1000g
1000i
1000h
1000d
1000j
1000k

Fig. 10F

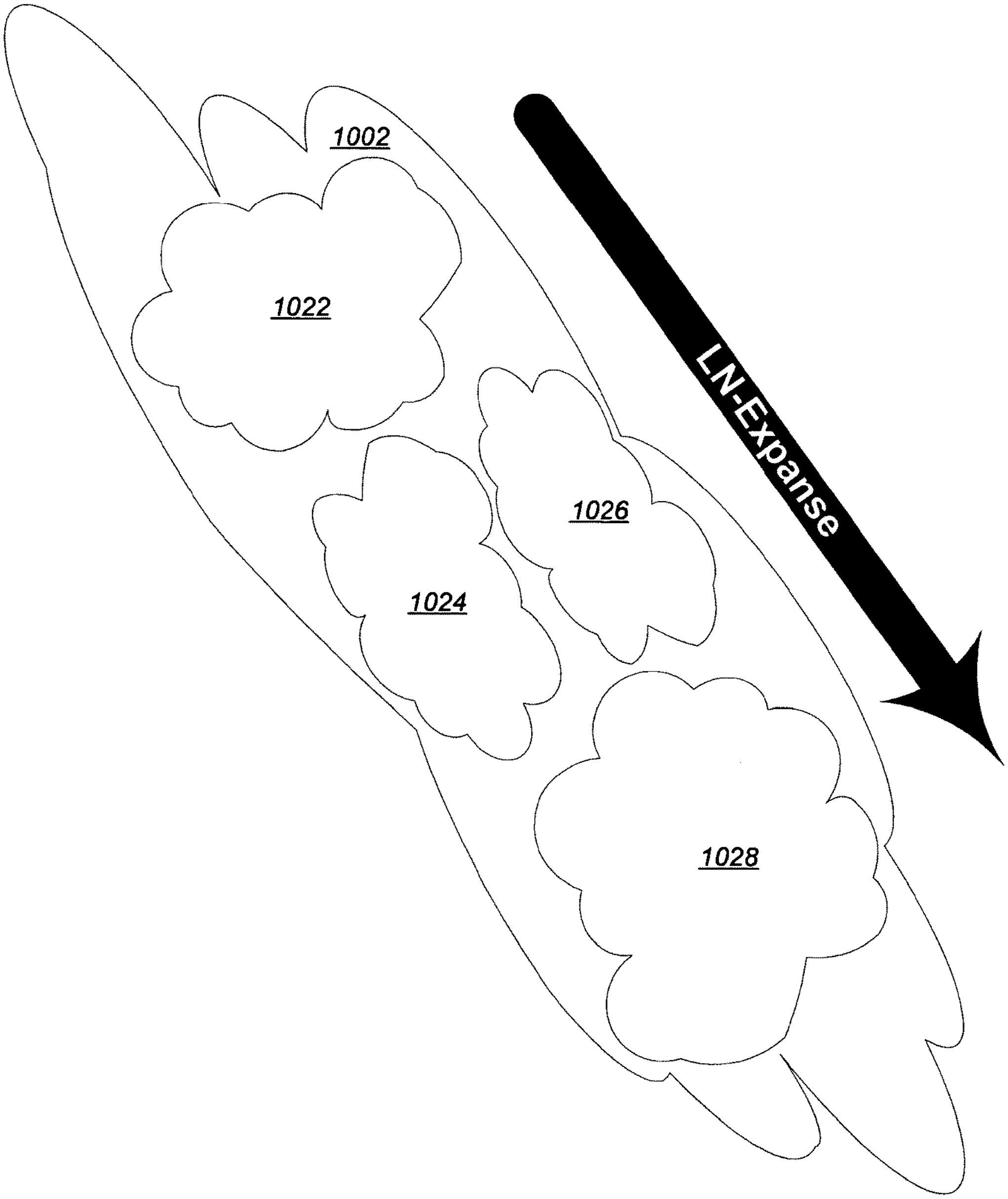


Fig. 10G

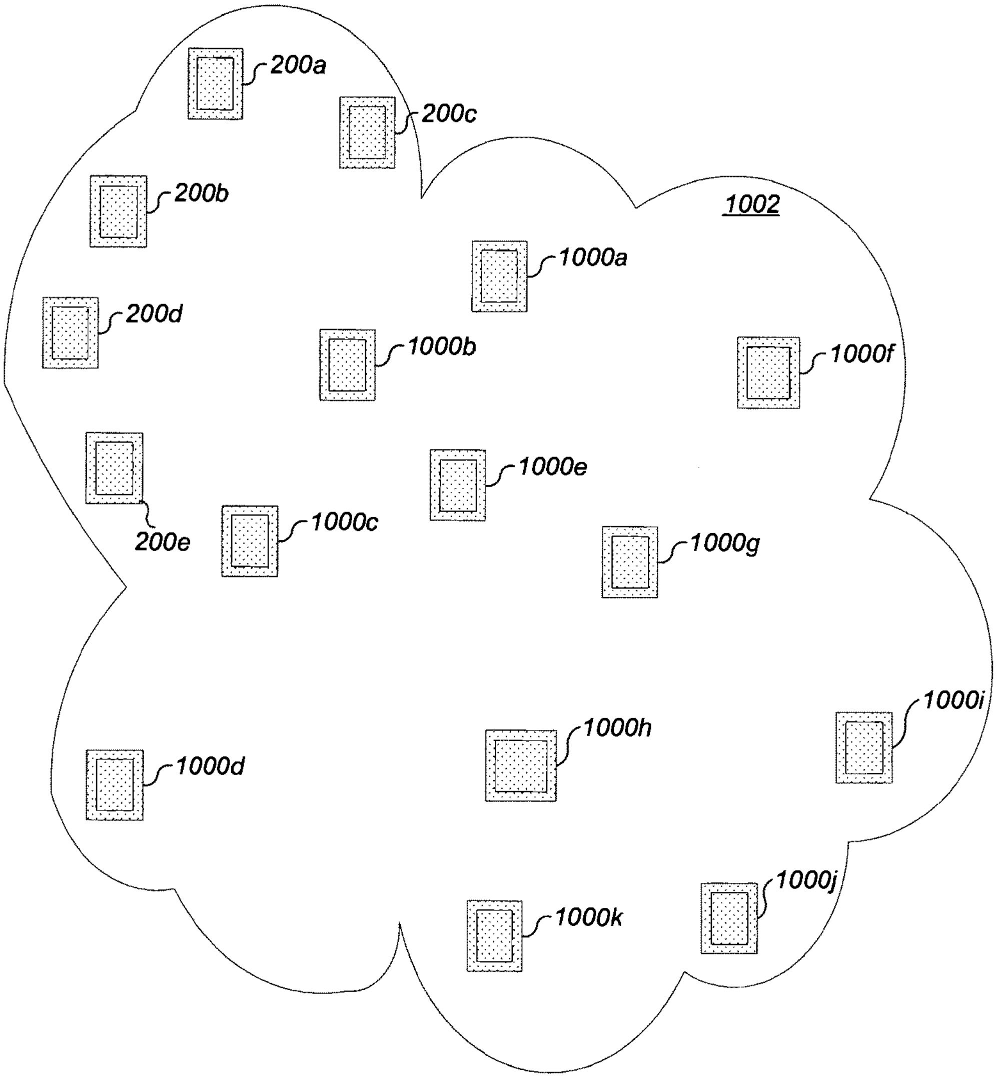

200a
200c
200b
1002
1000a
200d
1000b
1000f
1000e
200e
1000c
1000g
1000i
1000h
1000d
1000j
1000k

Fig. 10H

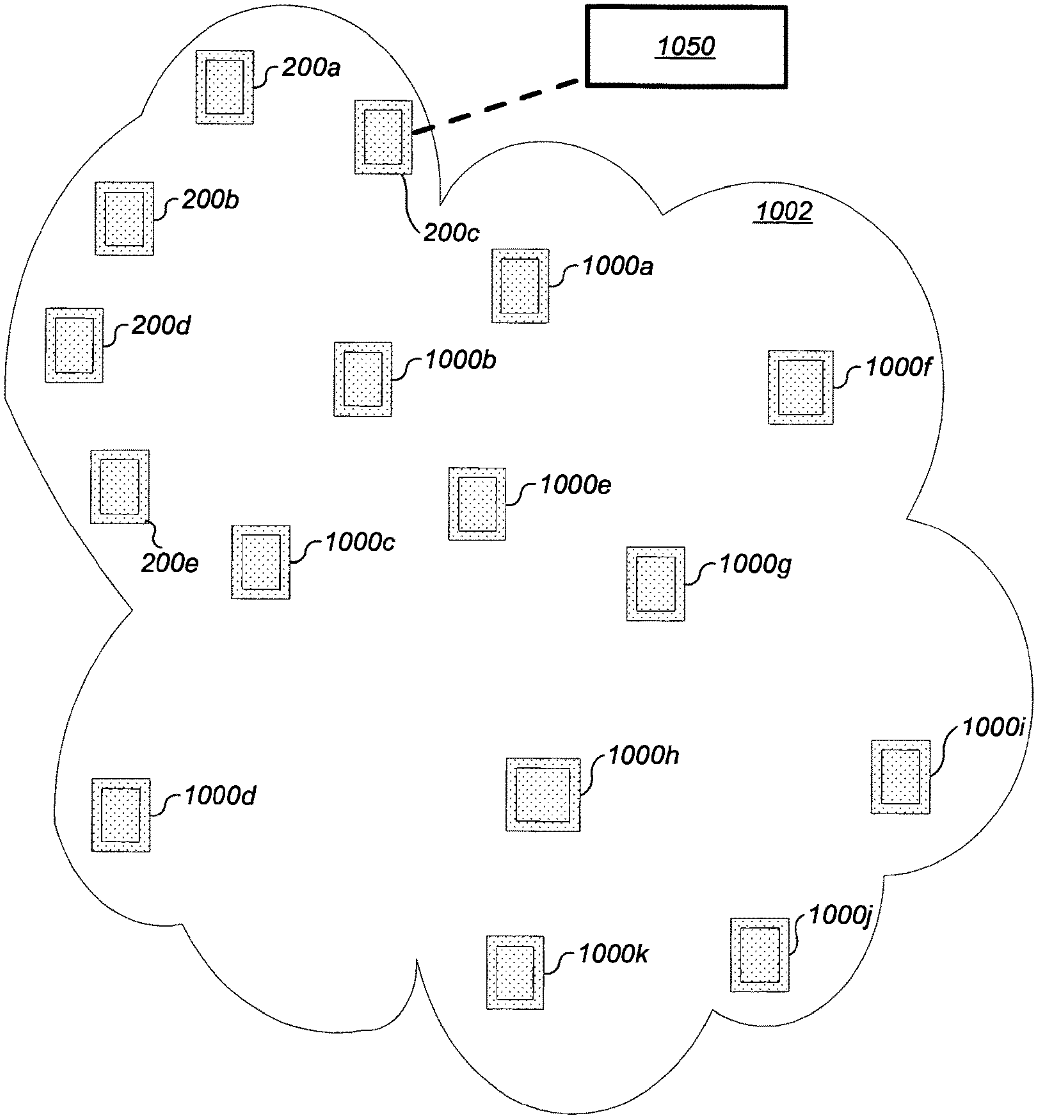

1050
200a
200b
200c
1002
1000a
200d
1000b
1000f
1000e
200e
1000c
1000g
1000i
1000h
1000d
1000j
1000k

Fig. 10I

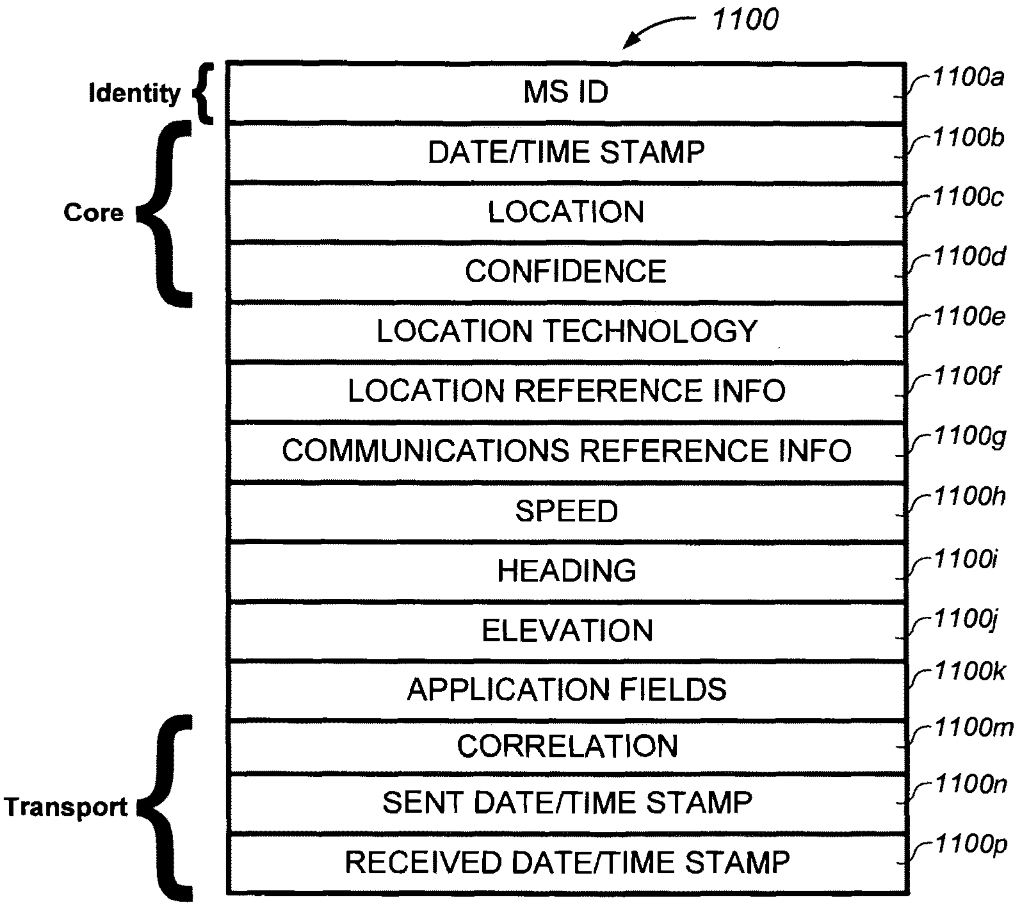

1100
Identity
Core
Transport
MS ID
DATE/TIME STAMP
LOCATION
CONFIDENCE
LOCATION TECHNOLOGY
LOCATION REFERENCE INFO
COMMUNICATIONS REFERENCE INFO
SPEED
HEADING
ELEVATION
APPLICATION FIELDS
CORRELATION
SENT DATE/TIME STAMP
RECEIVED DATE/TIME STAMP
1100a
1100b
1100c
1100d
1100e
1100f
1100g
1100h
1100i
1100j
1100k
1100m
1100n
1100p

Fig. 11A

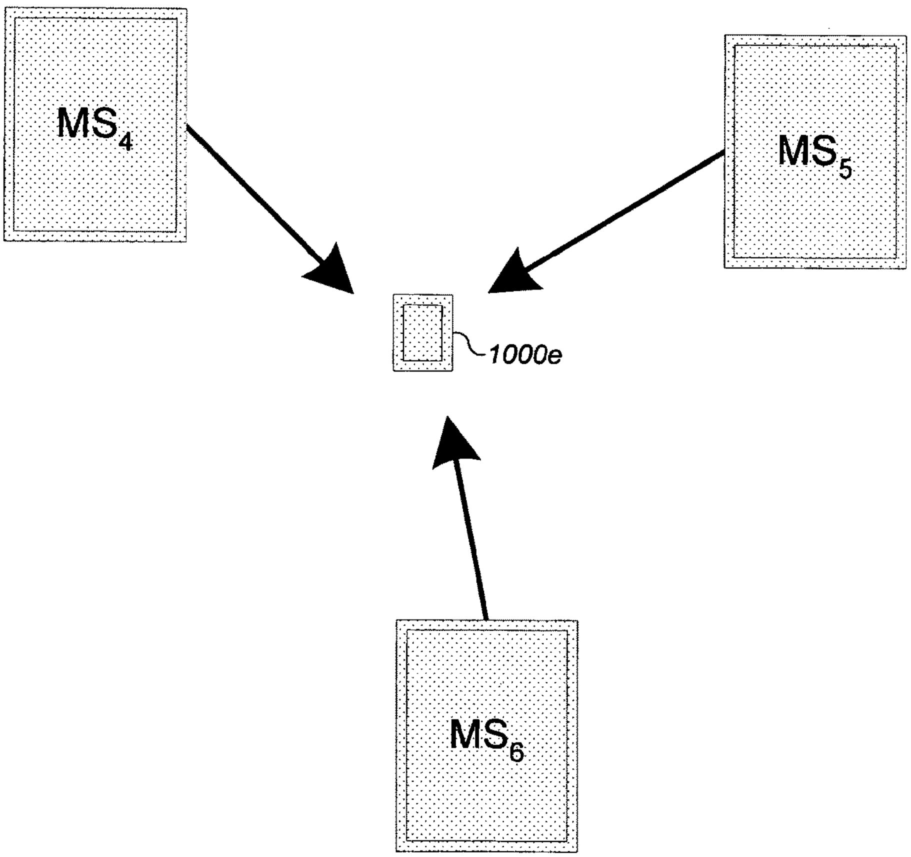


Fig. 11B

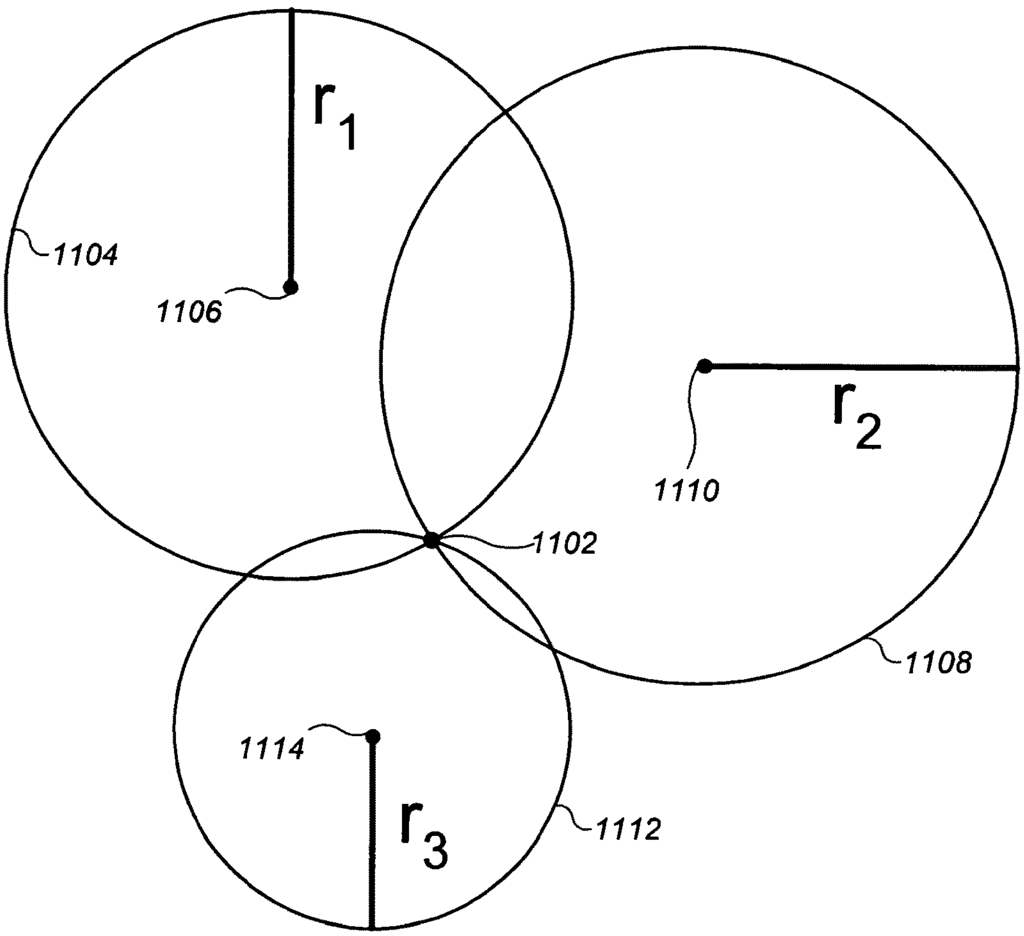


Fig. 11C

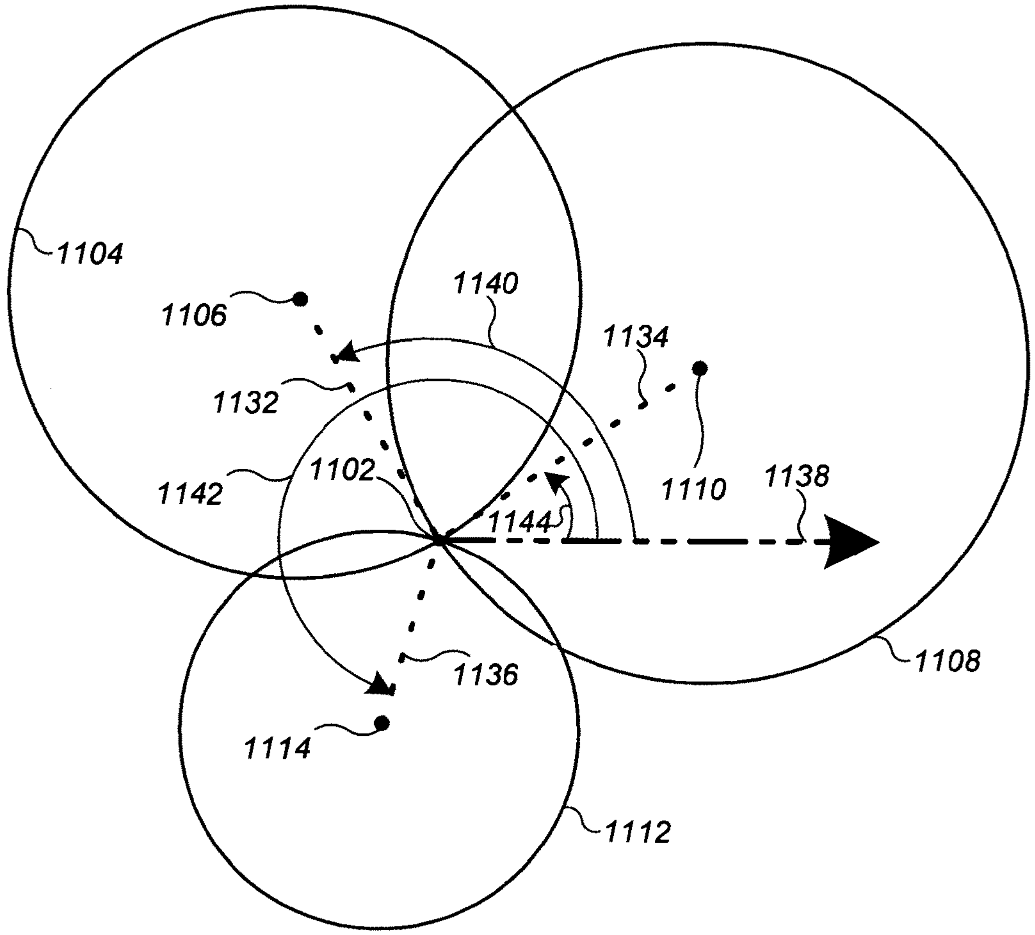

1104
1106
1140
1134
1132
1102
1142
1110
1138
1144
1108
1136
1114
1112

Fig. 11D

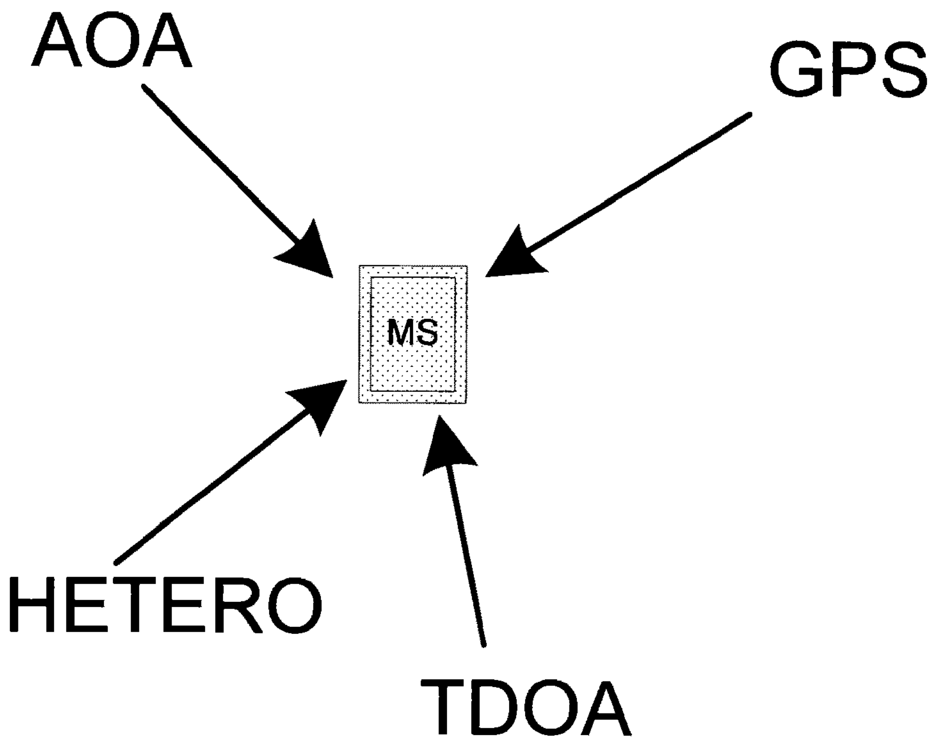


Fig. 11E

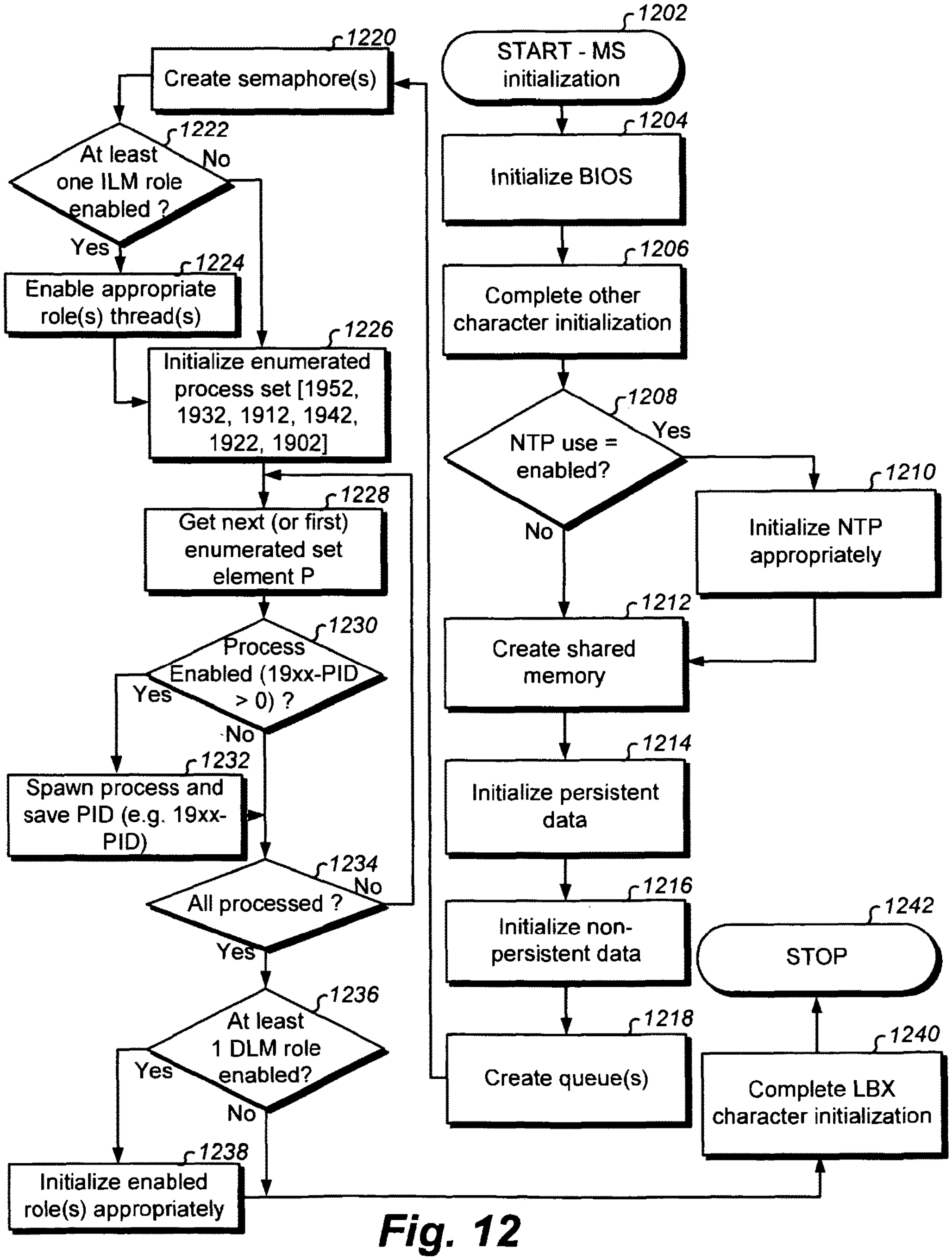

1202
START - MS initialization
1204
Initialize BIOS
1206
Complete other character initialization
1208
NTP use = enabled?
Yes
No
1210
Initialize NTP appropriately
1212
Create shared memory
1214
Initialize persistent data
1216
Initialize non-persistent data
1218
Create queue(s)
1220
Create semaphore(s)
1222
At least one ILM role enabled ?
No
Yes
1224
Enable appropriate role(s) thread(s)
1226
Initialize enumerated process set [1952, 1932, 1912, 1942, 1922, 1902]
1228
Get next (or first) enumerated set element P
1230
Process Enabled (19xx-PID > 0) ?
Yes
No
1232
Spawn process and save PID (e.g. 19xx-PID)
1234
All processed ?
No
Yes
1236
At least 1 DLM role enabled?
Yes
No
1238
Initialize enabled role(s) appropriately
1240
Complete LBX character initialization
1242
STOP

Fig. 12

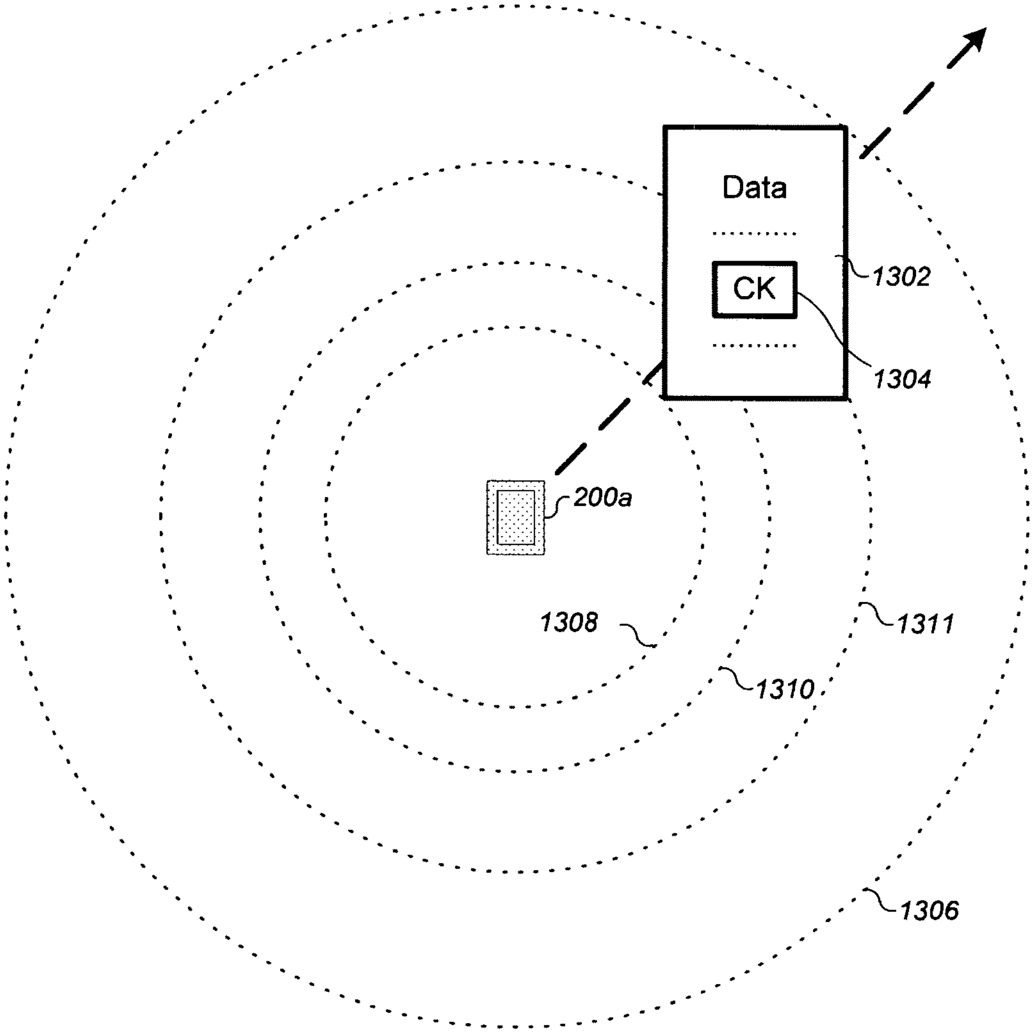


Fig. 13A

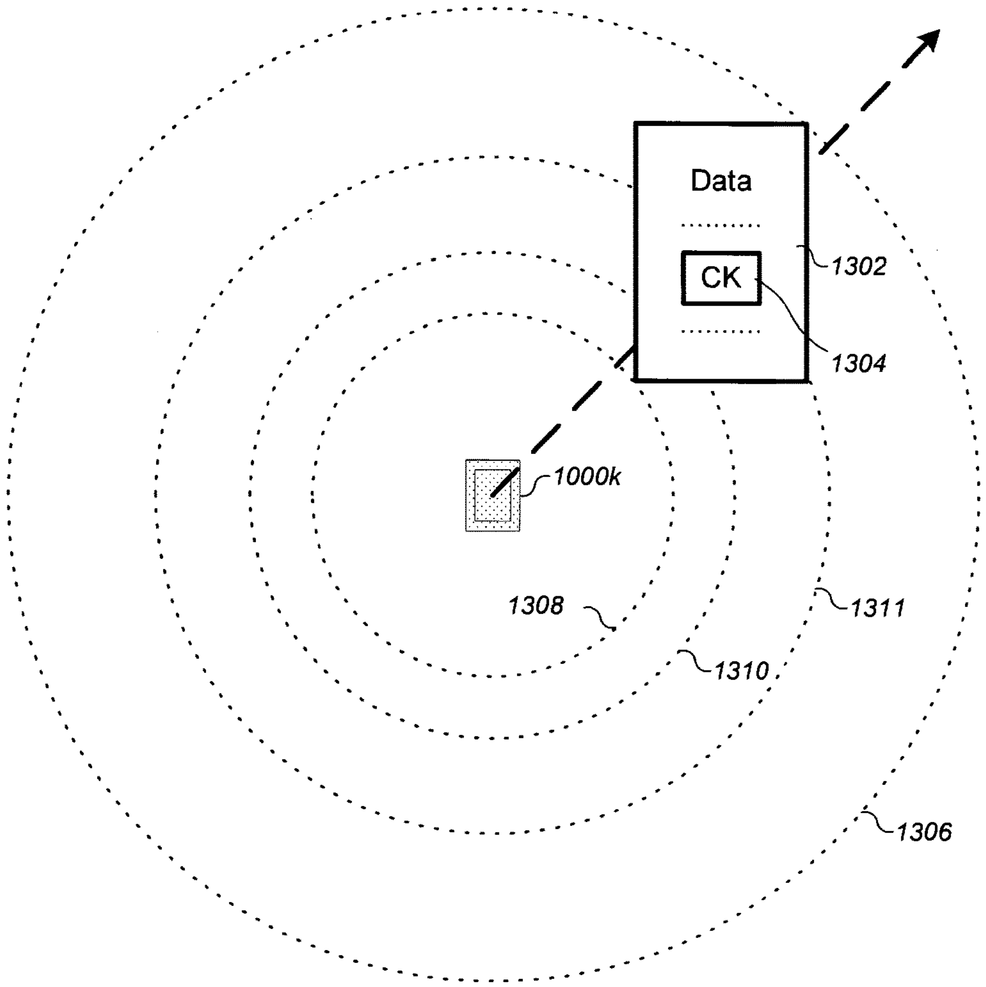


Fig. 13B

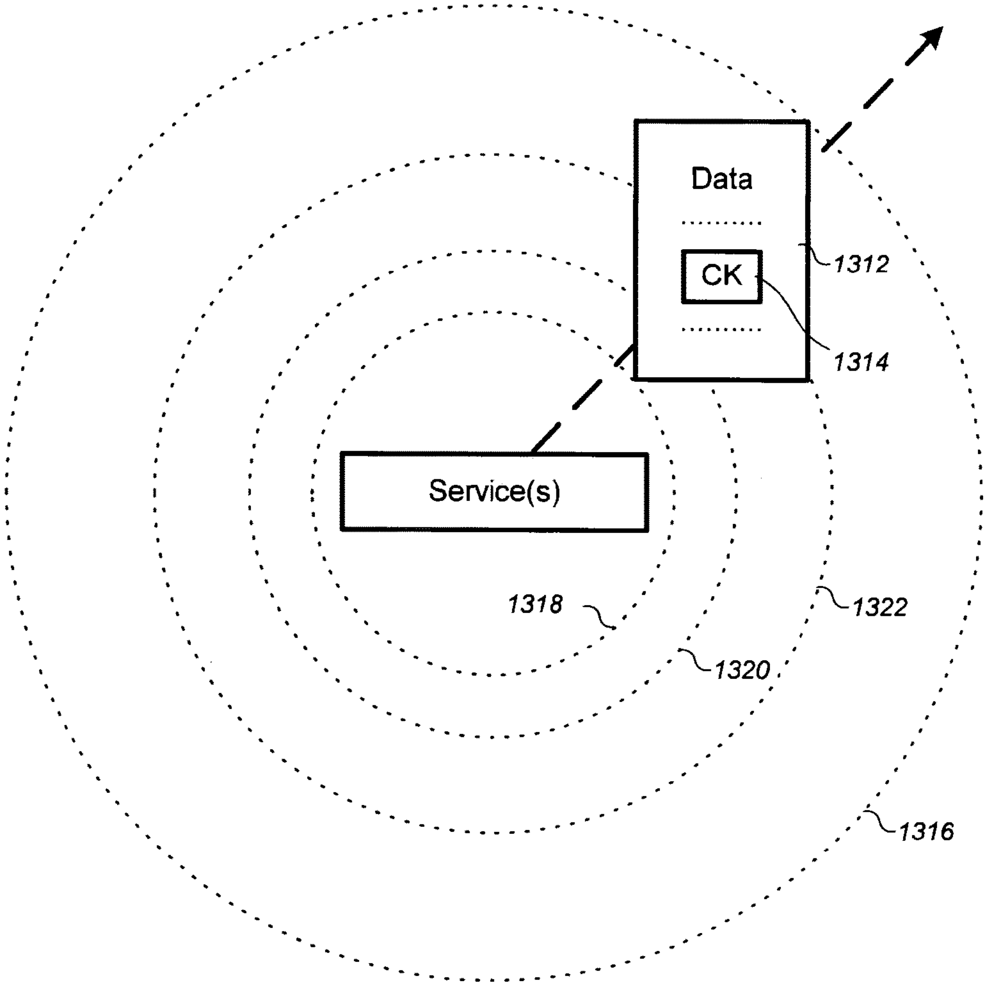


Fig. 13C

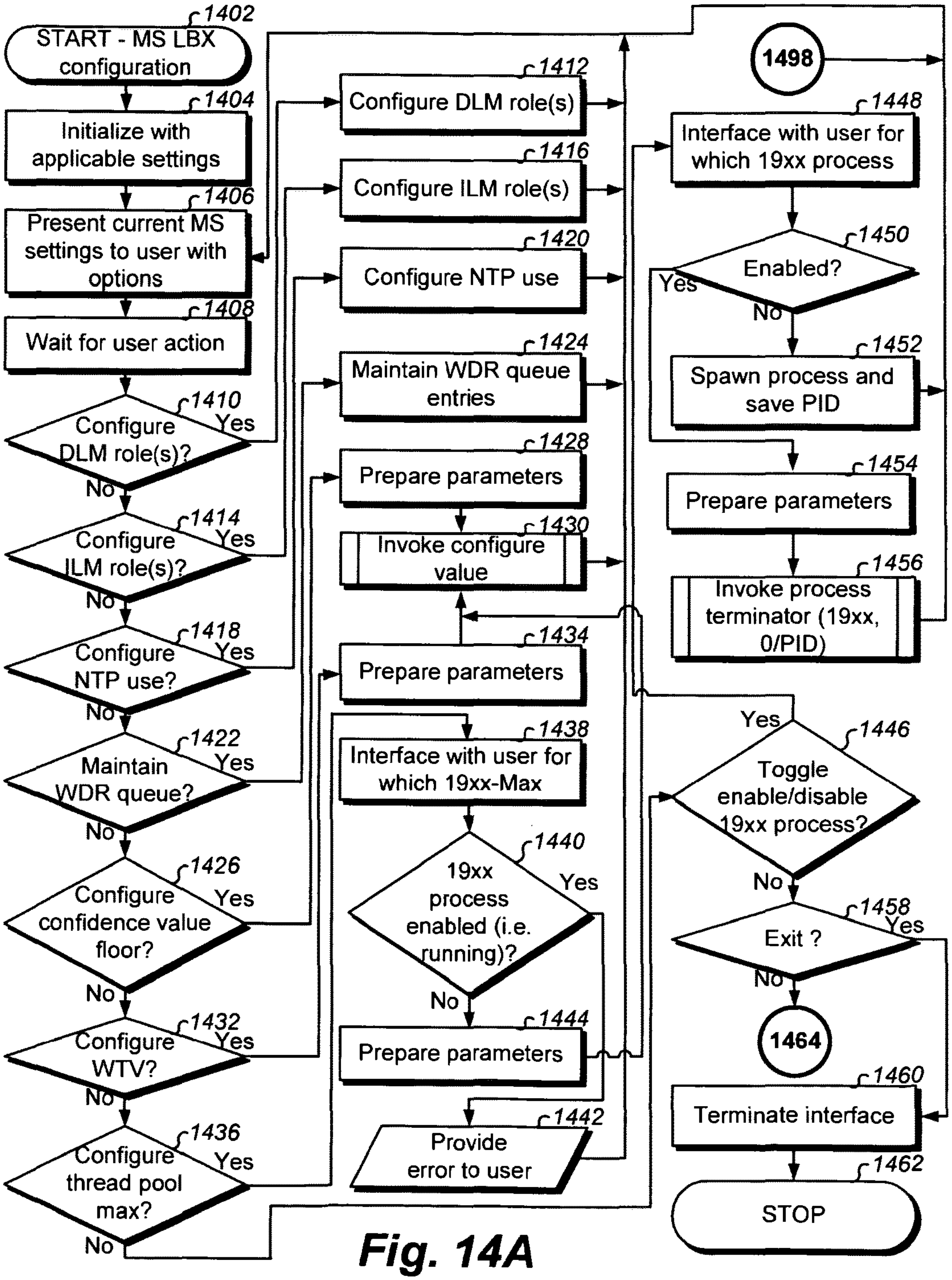

1402
START - MS LBX configuration
1404
Initialize with applicable settings
1406
Present current MS settings to user with options
1408
Wait for user action
1410
Configure DLM role(s)?
Yes
No
1414
Configure ILM role(s)?
Yes
No
1418
Configure NTP use?
Yes
No
1422
Maintain WDR queue?
Yes
No
1426
Configure confidence value floor?
Yes
No
1432
Configure WTV?
Yes
No
1436
Configure thread pool max?
Yes
No
1412
Configure DLM role(s)
1416
Configure ILM role(s)
1420
Configure NTP use
1424
Maintain WDR queue entries
1428
Prepare parameters
1430
Invoke configure value
1434
Prepare parameters
1438
Interface with user for which 19xx-Max
1440
19xx process enabled (i.e. running)?
Yes
No
1444
Prepare parameters
1442
Provide error to user
1498
1448
Interface with user for which 19xx process
1450
Enabled?
Yes
No
1452
Spawn process and save PID
1454
Prepare parameters
1456
Invoke process terminator (19xx, 0/PID)
Yes
1446
Toggle enable/disable 19xx process?
No
1458
Exit ?
Yes
No
1464
1460
Terminate interface
1462
STOP

Fig. 14A

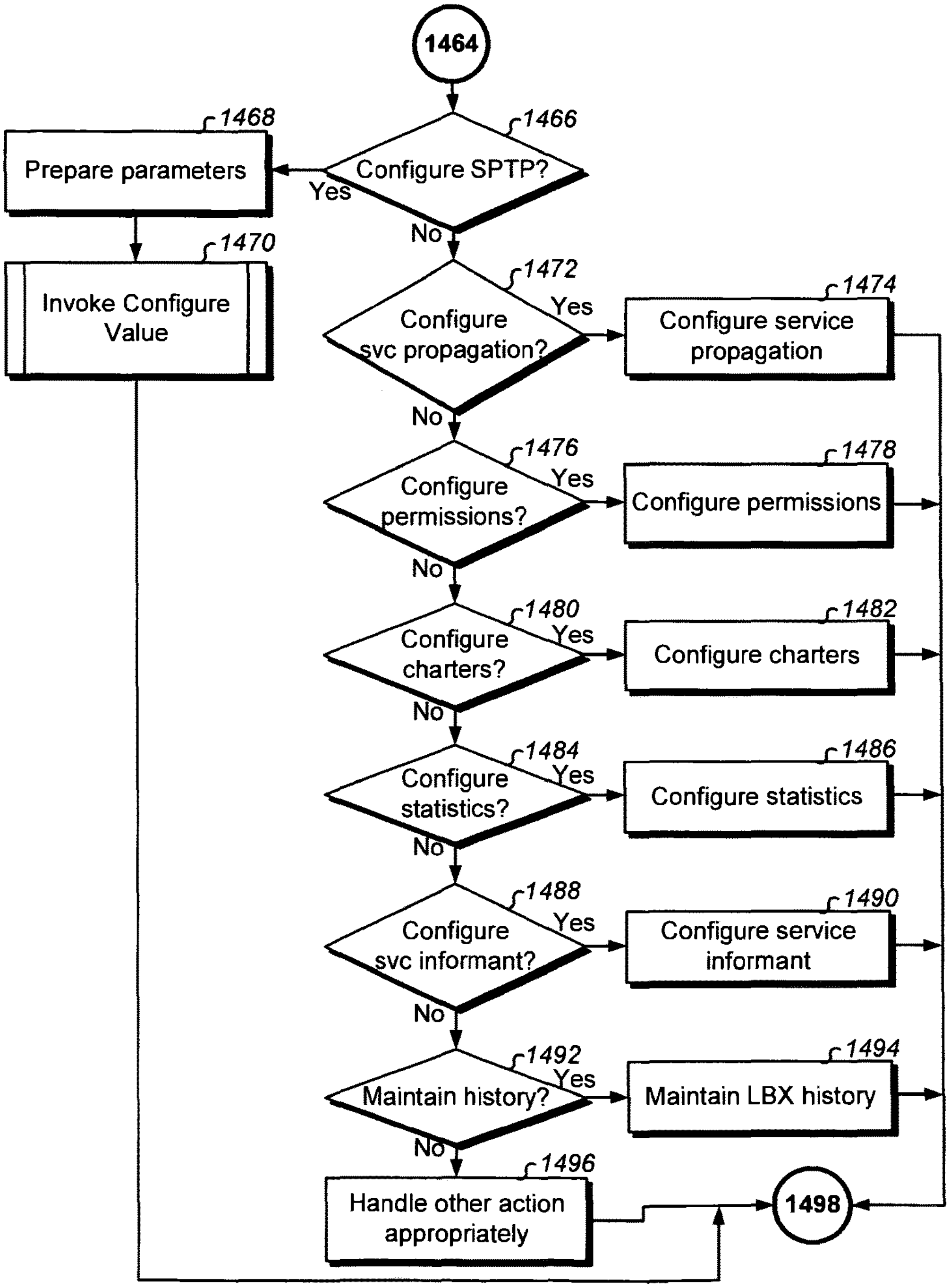

1464
1466
Configure SPTP?
Yes
No
1468
Prepare parameters
1470
Invoke Configure Value
1472
Configure svc propagation?
Yes
No
1474
Configure service propagation
1476
Configure permissions?
Yes
No
1478
Configure permissions
1480
Configure charters?
Yes
No
1482
Configure charters
1484
Configure statistics?
Yes
No
1486
Configure statistics
1488
Configure svc informant?
Yes
No
1490
Configure service informant
1492
Maintain history?
Yes
No
1494
Maintain LBX history
1496
Handle other action appropriately
1498

Fig. 14B

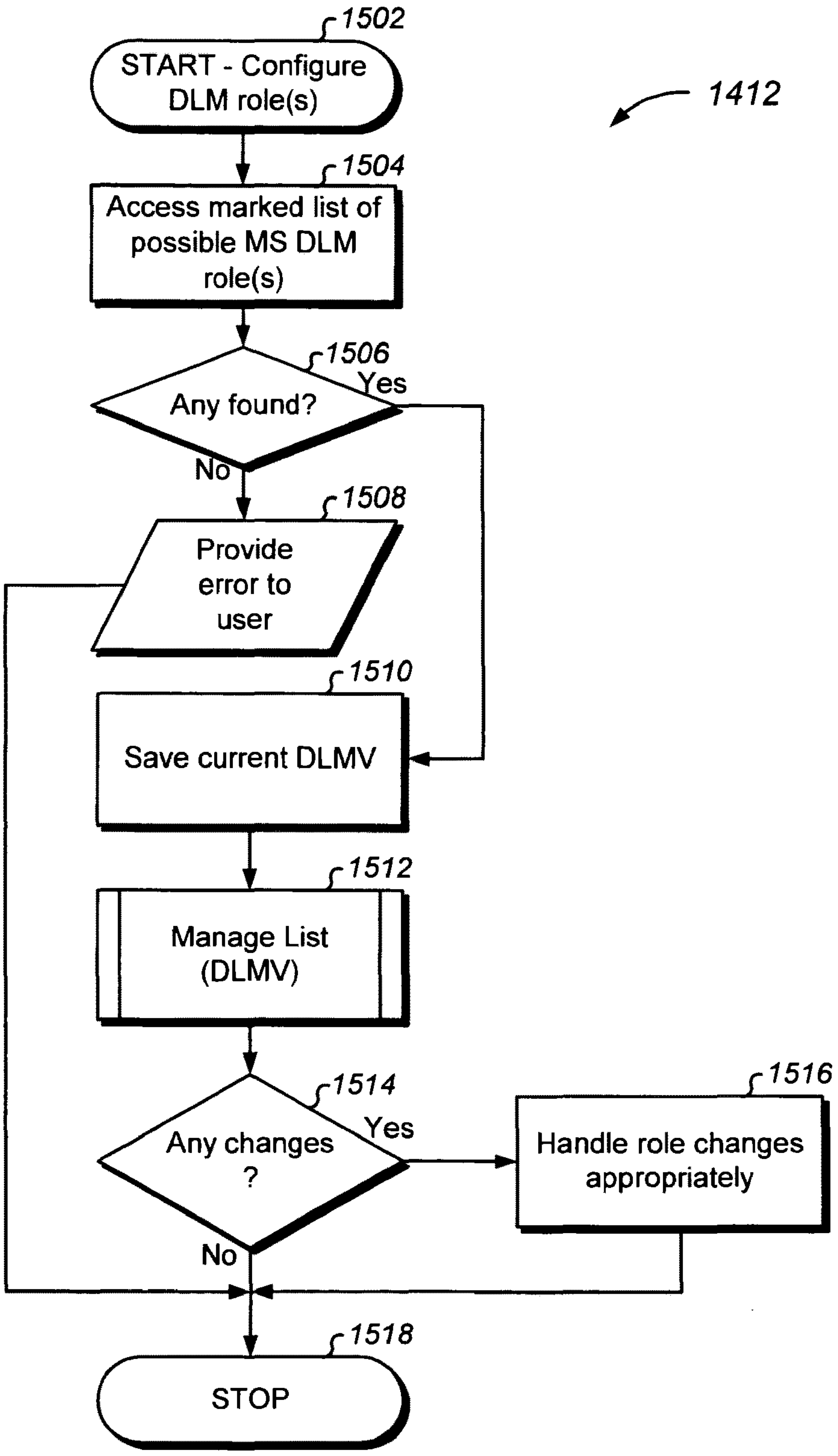

1412
1502
START - Configure DLM role(s)
1504
Access marked list of possible MS DLM role(s)
1506
Any found?
Yes
No
1508
Provide error to user
1510
Save current DLMV
1512
Manage List (DLMV)
1514
Any changes ?
Yes
No
1516
Handle role changes appropriately
1518
STOP

Fig. 15A

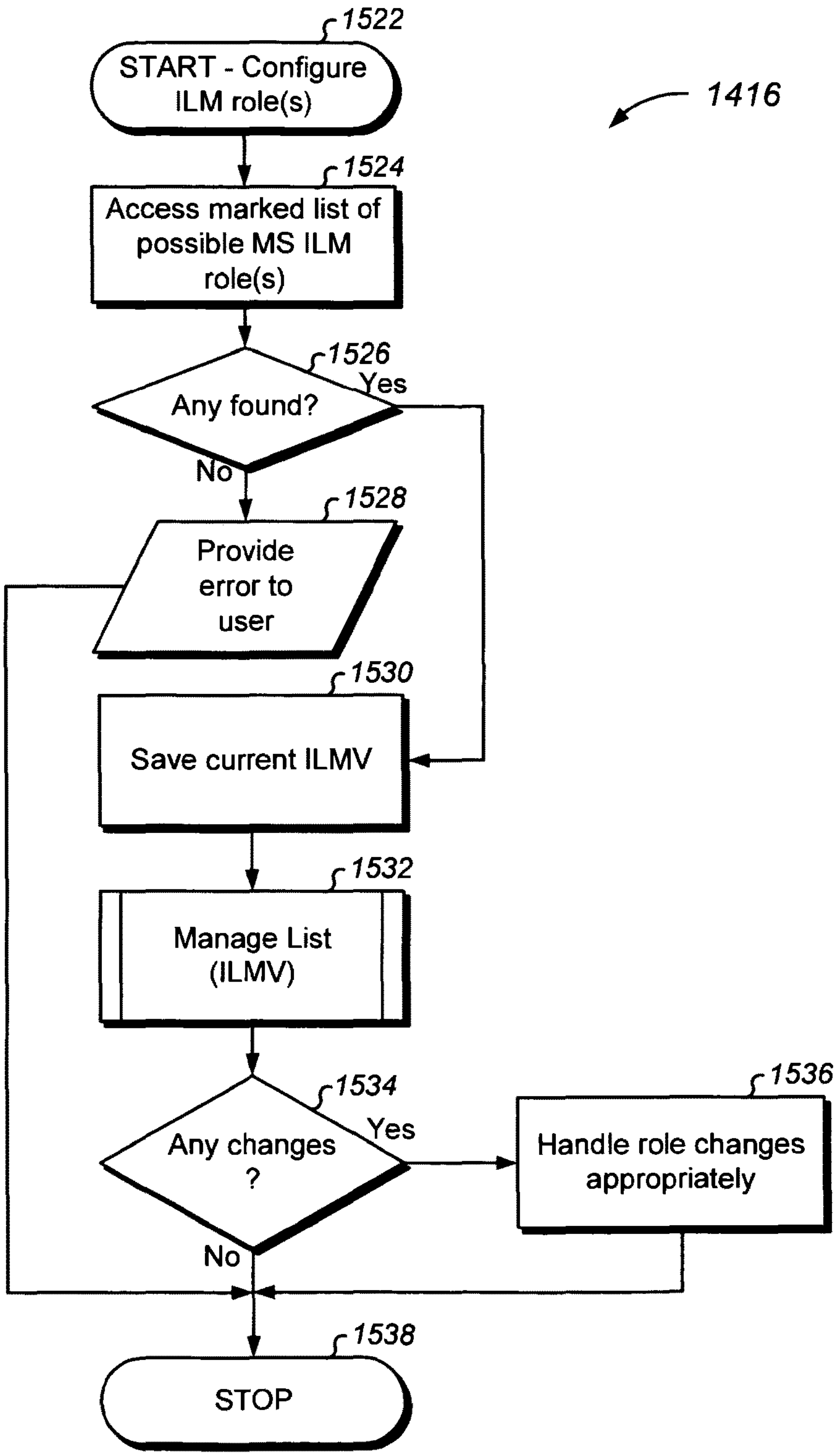

1416
1522
START - Configure ILM role(s)
1524
Access marked list of possible MS ILM role(s)
1526
Any found?
Yes
No
1528
Provide error to user
1530
Save current ILMV
1532
Manage List (ILMV)
1534
Any changes ?
Yes
No
1536
Handle role changes appropriately
1538
STOP

Fig. 15B

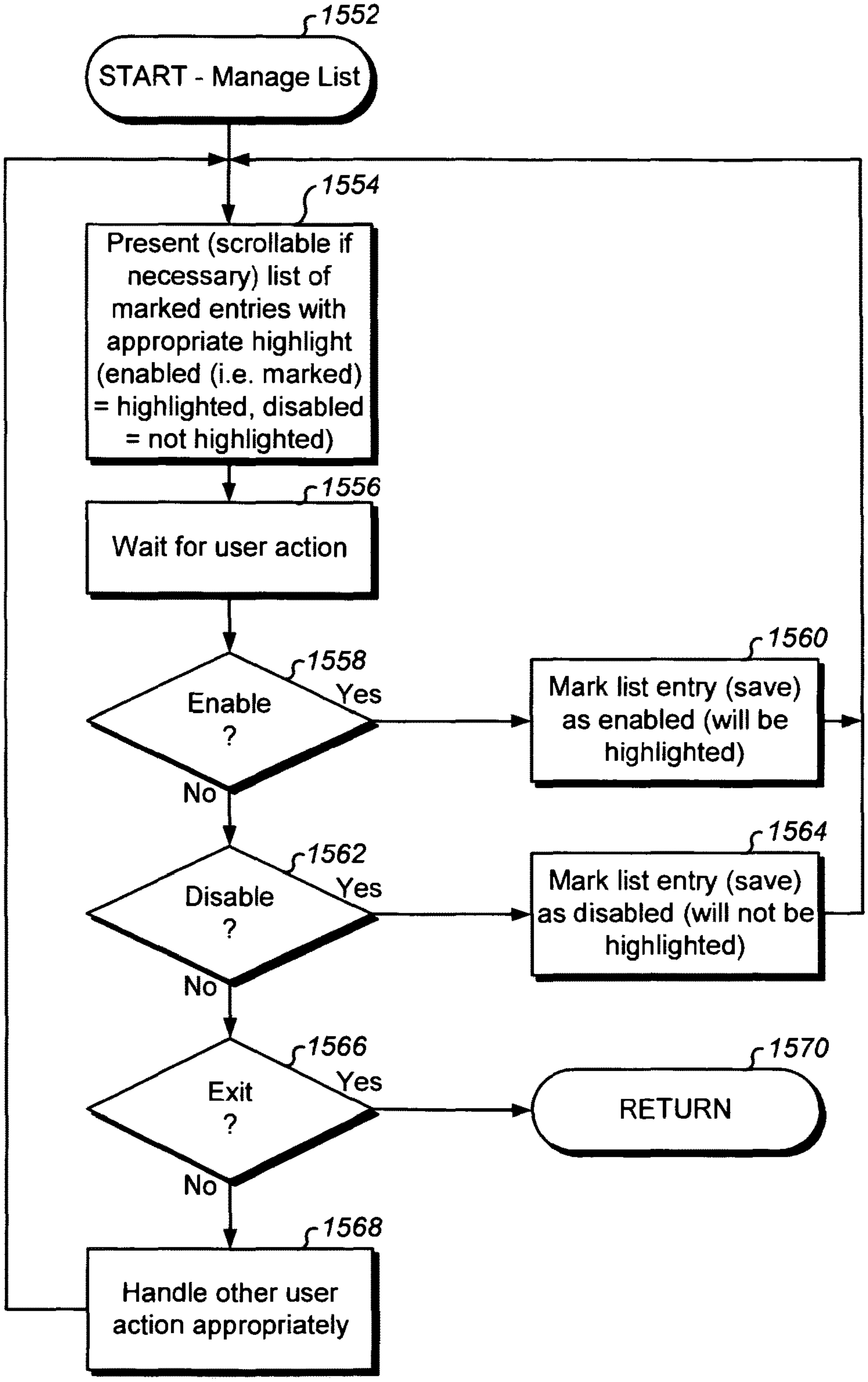

1552
START - Manage List
1554
Present (scrollable if necessary) list of marked entries with appropriate highlight (enabled (i.e. marked) = highlighted, disabled = not highlighted)
1556
Wait for user action
1558
Enable ?
Yes
No
1560
Mark list entry (save) as enabled (will be highlighted)
1562
Disable ?
Yes
No
1564
Mark list entry (save) as disabled (will not be highlighted)
1566
Exit ?
Yes
No
1570
RETURN
1568
Handle other user action appropriately

Fig. 15C

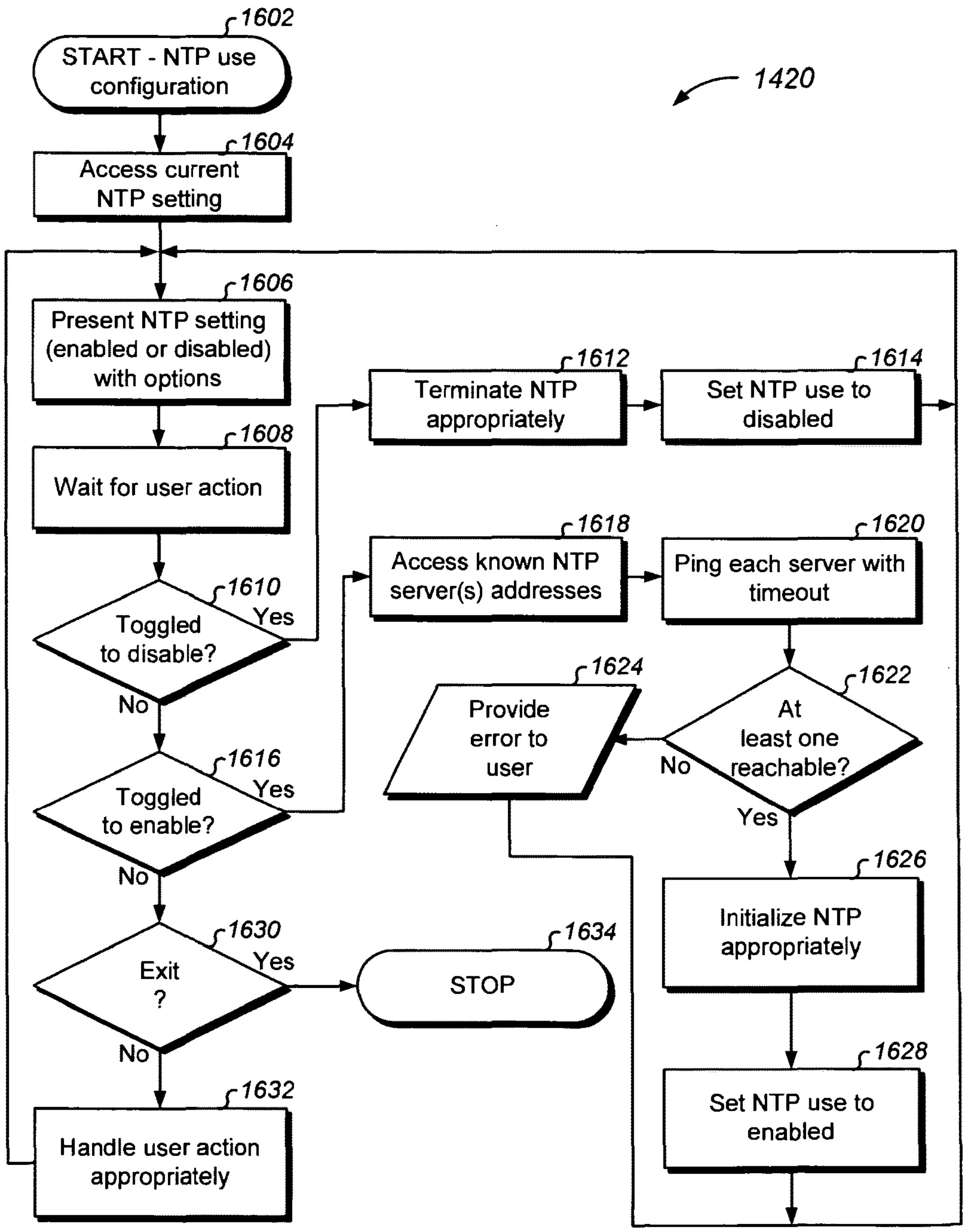

1602
START - NTP use configuration
1420
1604
Access current NTP setting
1606
Present NTP setting (enabled or disabled) with options
1612
Terminate NTP appropriately
1614
Set NTP use to disabled
1608
Wait for user action
1618
Access known NTP server(s) addresses
1620
Ping each server with timeout
1610
Toggled to disable?
Yes
No
1624
Provide error to user
1622
At least one reachable?
No
Yes
1616
Toggled to enable?
Yes
No
1626
Initialize NTP appropriately
1630
Exit ?
Yes
No
1634
STOP
1628
Set NTP use to enabled
1632
Handle user action appropriately

Fig. 16

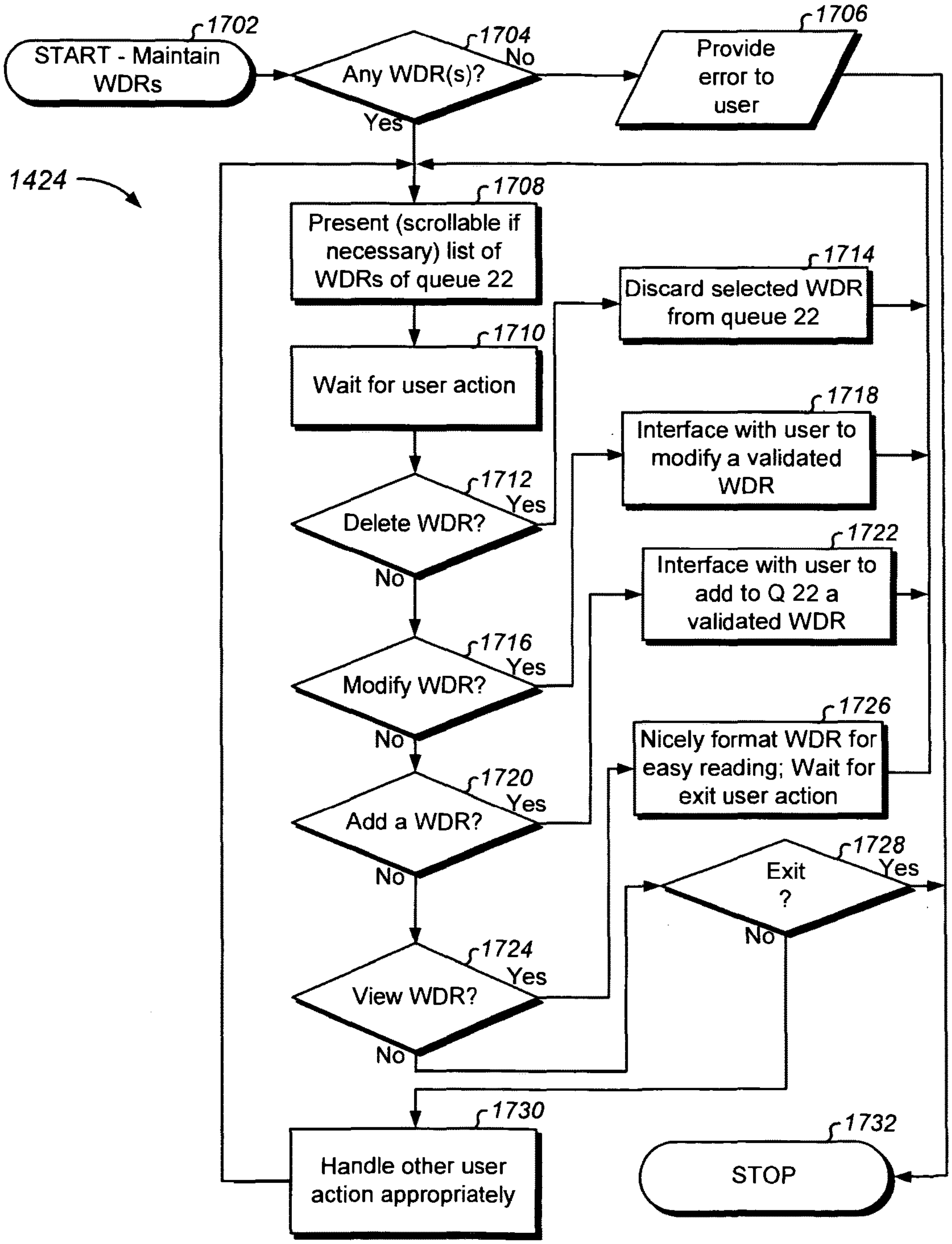

1702
START - Maintain WDRs
1704
Any WDR(s)?
No
Yes
1706
Provide error to user
1424
1708
Present (scrollable if necessary) list of WDRs of queue 22
1710
Wait for user action
1712
Delete WDR?
Yes
No
1714
Discard selected WDR from queue 22
1716
Modify WDR?
Yes
No
1718
Interface with user to modify a validated WDR
1720
Add a WDR?
Yes
No
1722
Interface with user to add to Q 22 a validated WDR
1724
View WDR?
Yes
No
1726
Nicely format WDR for easy reading; Wait for exit user action
1728
Exit ?
Yes
No
1730
Handle other user action appropriately
1732
STOP

Fig. 17

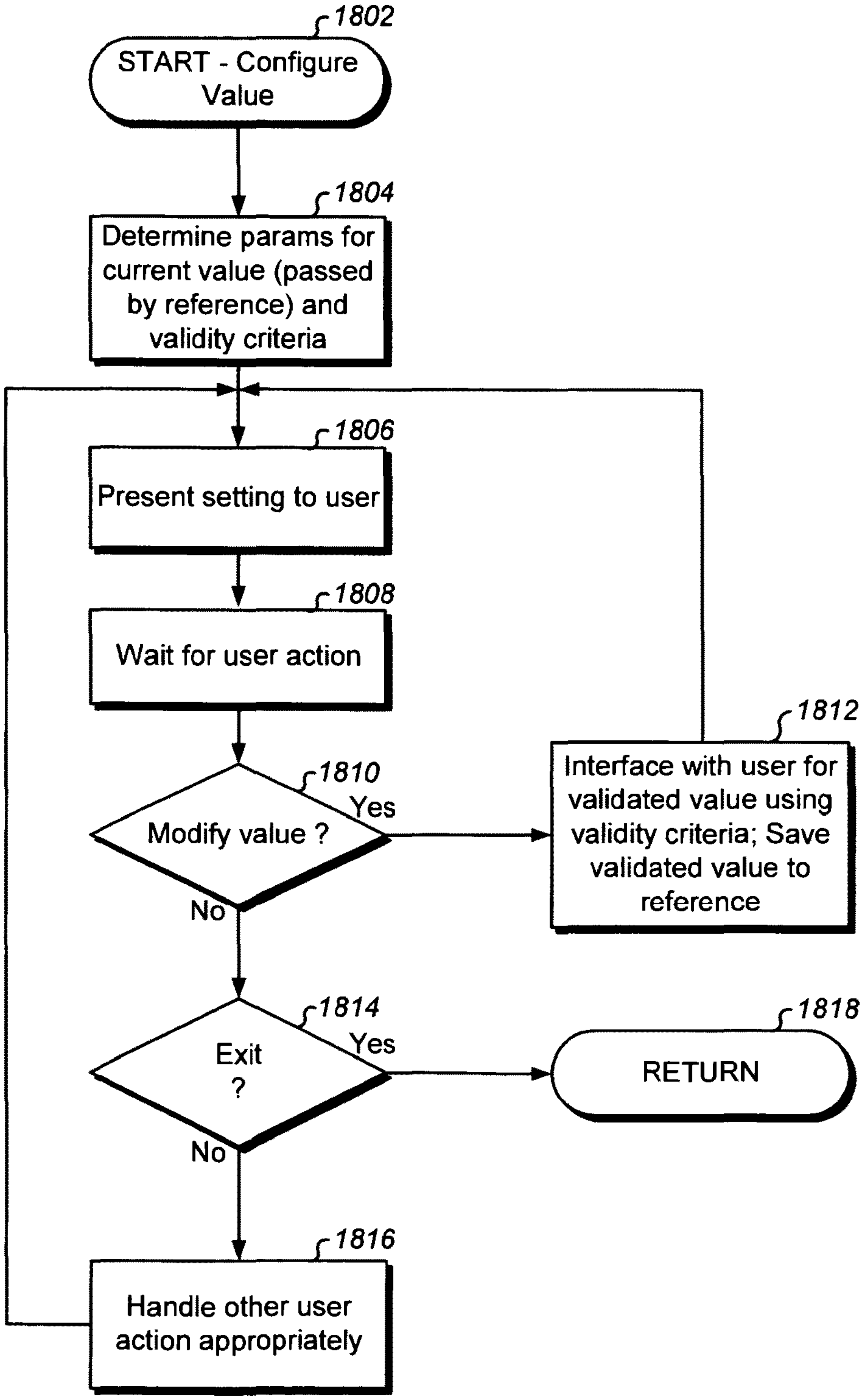

1802
START - Configure Value
1804
Determine params for current value (passed by reference) and validity criteria
1806
Present setting to user
1808
Wait for user action
1810
Modify value ?
Yes
No
1812
Interface with user for validated value using validity criteria; Save validated value to reference
1814
Exit ?
Yes
No
1818
RETURN
1816
Handle other user action appropriately

Fig. 18

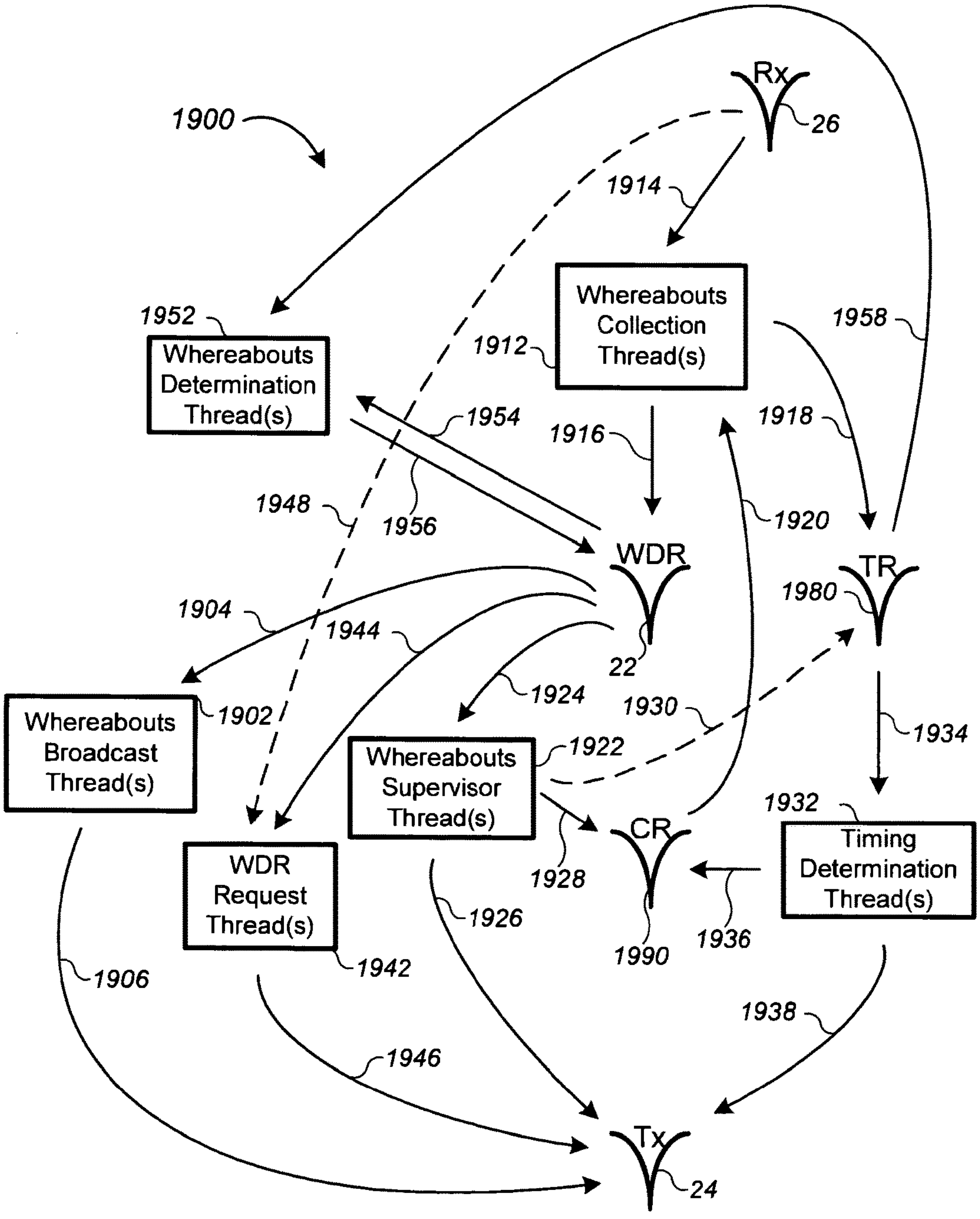

1900
Rx
26
1914
Whereabouts Collection Thread(s)
1912
1952
Whereabouts Determination Thread(s)
1958
1954
1916
1918
1948
1956
1920
WDR
TR
1980
1904
1944
22
1924
1930
1934
Whereabouts Broadcast Thread(s)
1902
Whereabouts Supervisor Thread(s)
1922
1932
CR
Timing Determination Thread(s)
WDR Request Thread(s)
1928
1926
1936
1990
1906
1942
1938
1946
Tx
24

Fig. 19

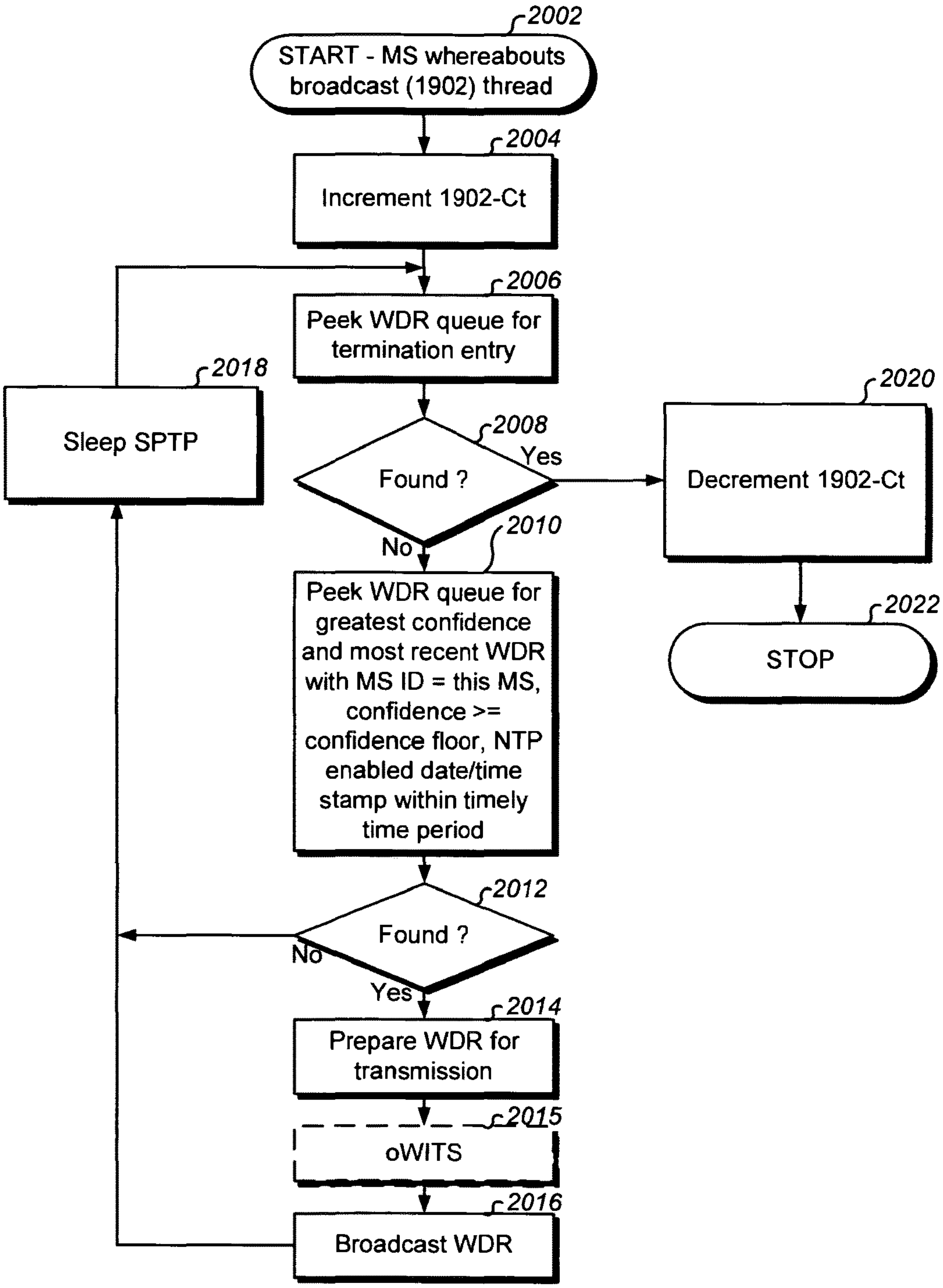

2002
START - MS whereabouts broadcast (1902) thread
2004
Increment 1902-Ct
2006
Peek WDR queue for termination entry
2008
Found ?
Yes
No
2020
Decrement 1902-Ct
2022
STOP
2010
Peek WDR queue for greatest confidence and most recent WDR with MS ID = this MS, confidence >= confidence floor, NTP enabled date/time stamp within timely time period
2012
Found ?
No
Yes
2014
Prepare WDR for transmission
2015
oWITS
2016
Broadcast WDR
2018
Sleep SPTP

Fig. 20

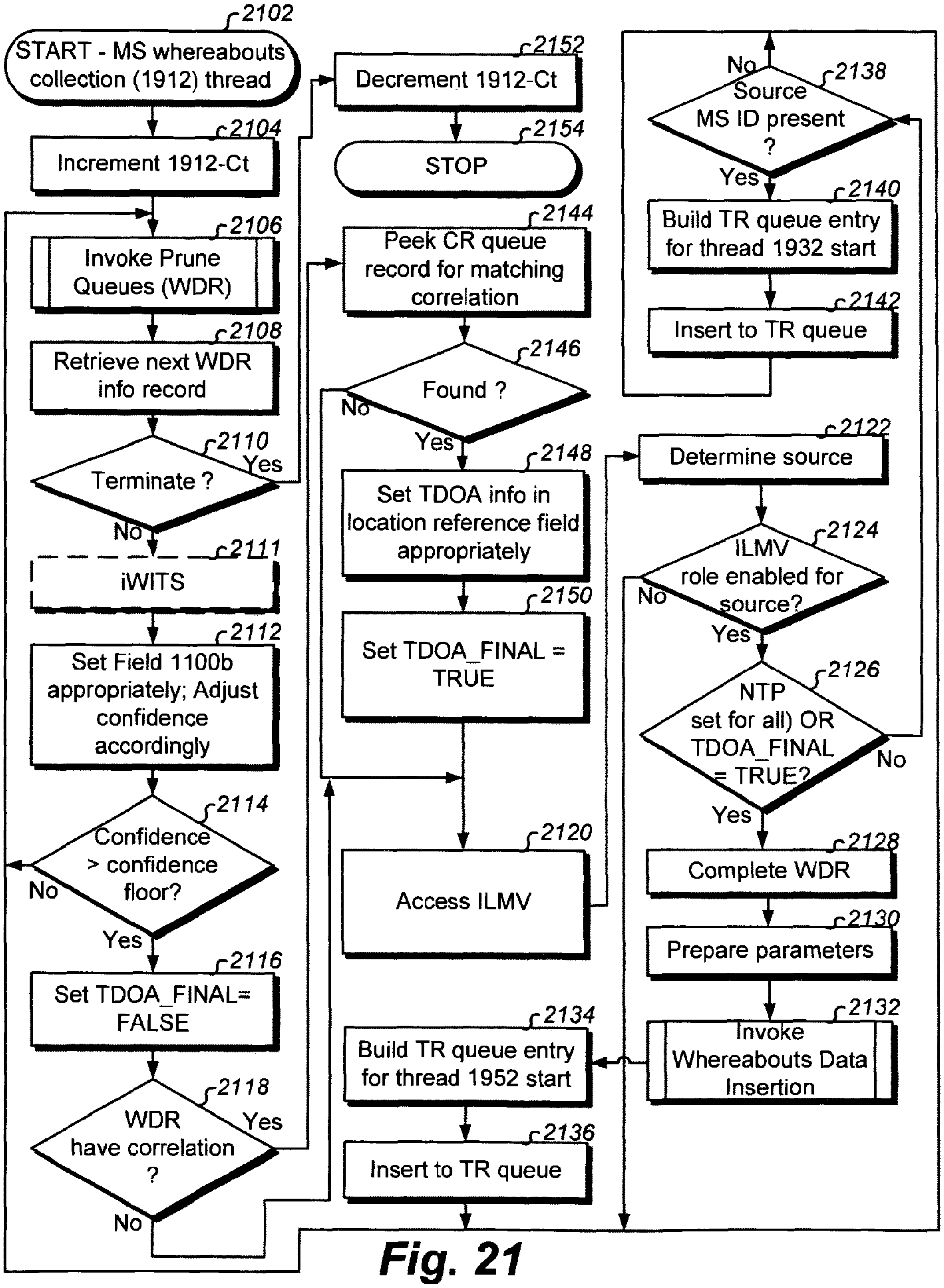

START - MS whereabouts collection (1912) thread
2102
Increment 1912-Ct
2104
Invoke Prune Queues (WDR)
2106
Retrieve next WDR info record
2108
Terminate ?
2110
Yes
No
iWITS
2111
Set Field 1100b appropriately; Adjust confidence accordingly
2112
Confidence > confidence floor?
2114
No
Yes
Set TDOA_FINAL= FALSE
2116
WDR have correlation ?
2118
Yes
No
Decrement 1912-Ct
2152
STOP
2154
Peek CR queue record for matching correlation
2144
Found ?
2146
No
Yes
Set TDOA info in location reference field appropriately
2148
Set TDOA_FINAL = TRUE
2150
Access ILMV
2120
Build TR queue entry for thread 1952 start
2134
Insert to TR queue
2136
Source MS ID present ?
2138
No
Yes
Build TR queue entry for thread 1932 start
2140
Insert to TR queue
2142
Determine source
2122
ILMV role enabled for source?
2124
No
Yes
NTP set for all) OR TDOA_FINAL = TRUE?
2126
No
Yes
Complete WDR
2128
Prepare parameters
2130
Invoke Whereabouts Data Insertion
2132

Fig. 21

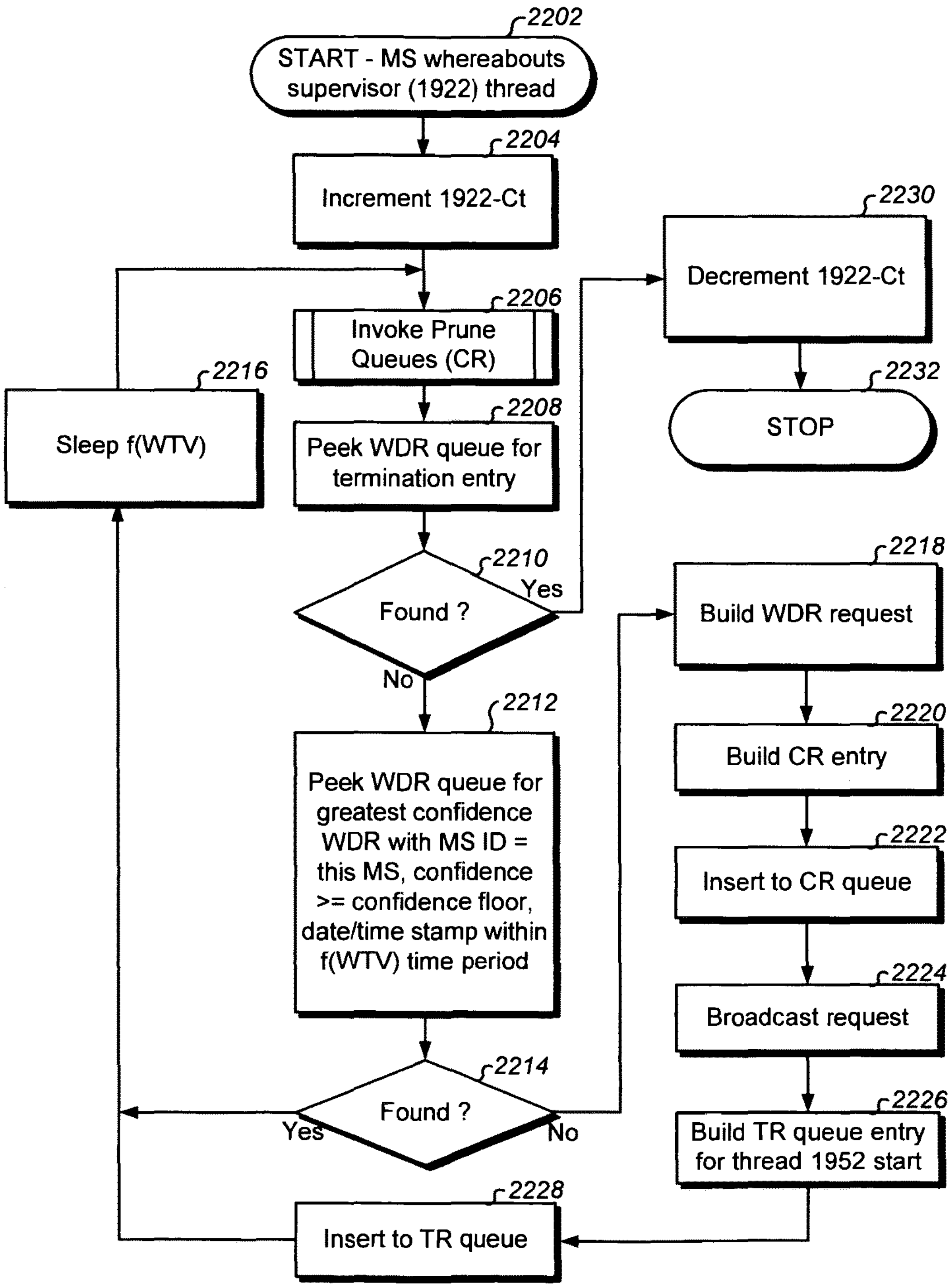

2202
START - MS whereabouts supervisor (1922) thread
2204
Increment 1922-Ct
2206
Invoke Prune Queues (CR)
2208
Peek WDR queue for termination entry
2210
Found ?
Yes
No
2212
Peek WDR queue for greatest confidence WDR with MS ID = this MS, confidence >= confidence floor, date/time stamp within f(WTV) time period
2214
Found ?
Yes
No
2216
Sleep f(WTV)
2228
Insert to TR queue
2230
Decrement 1922-Ct
2232
STOP
2218
Build WDR request
2220
Build CR entry
2222
Insert to CR queue
2224
Broadcast request
2226
Build TR queue entry for thread 1952 start

Fig. 22

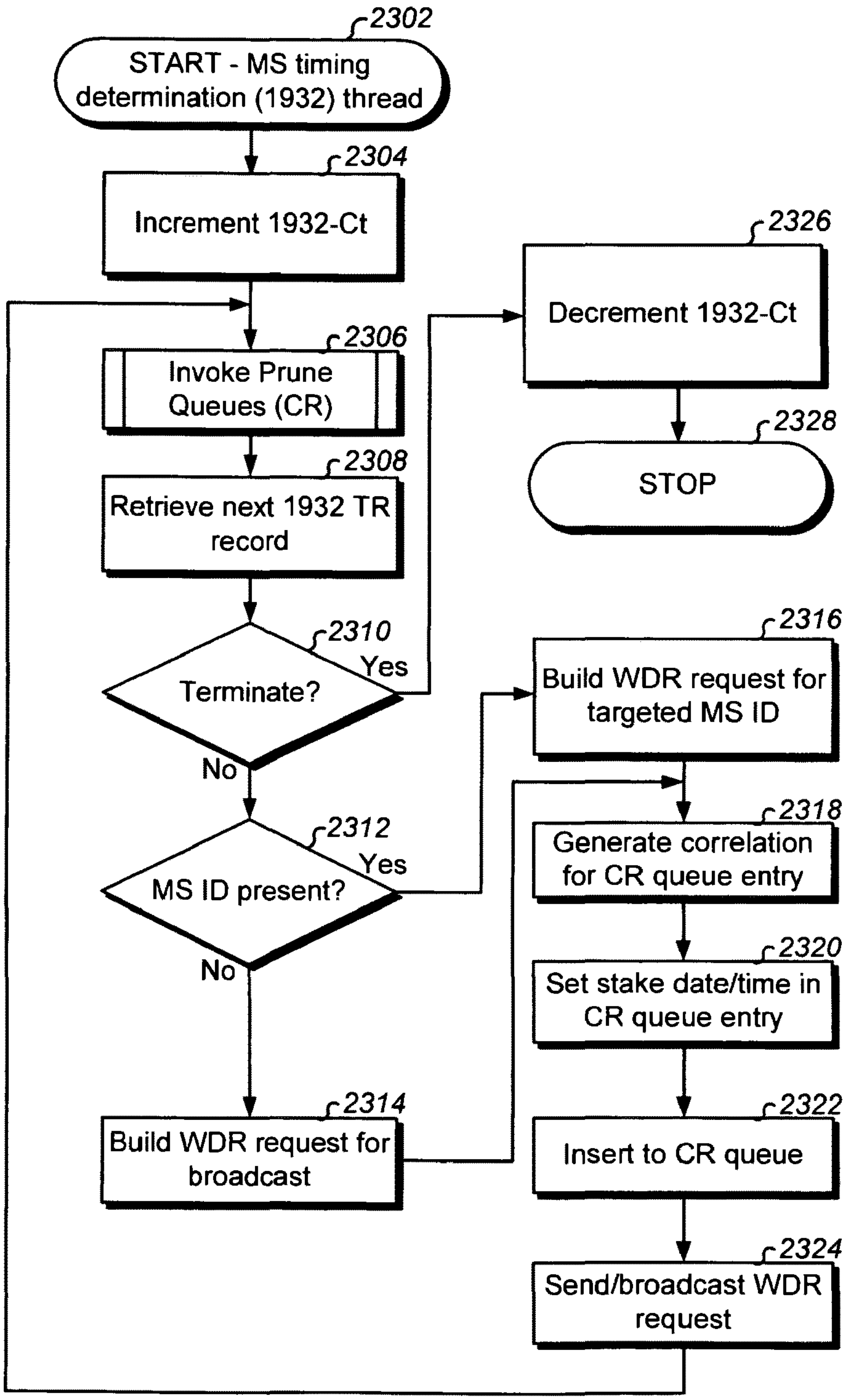

2302
START - MS timing determination (1932) thread
2304
Increment 1932-Ct
2306
Invoke Prune Queues (CR)
2308
Retrieve next 1932 TR record
2310
Terminate?
Yes
No
2312
MS ID present?
Yes
No
2314
Build WDR request for broadcast
2326
Decrement 1932-Ct
2328
STOP
2316
Build WDR request for targeted MS ID
2318
Generate correlation for CR queue entry
2320
Set stake date/time in CR queue entry
2322
Insert to CR queue
2324
Send/broadcast WDR request

Fig. 23

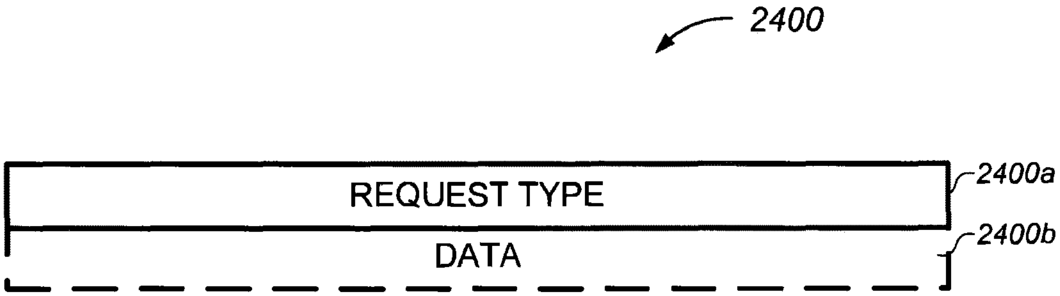


Fig. 24A

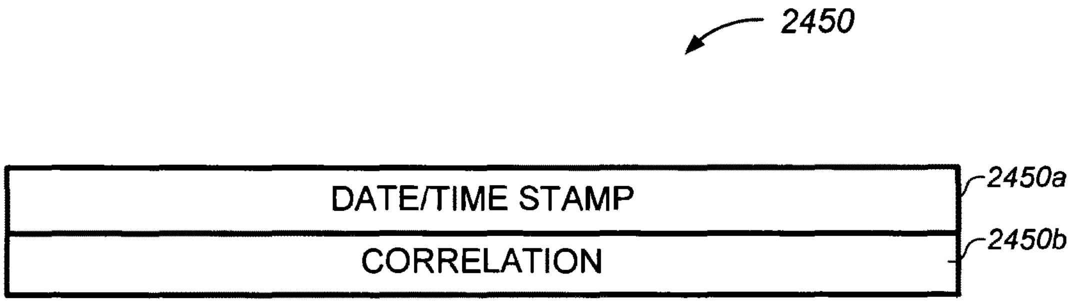


Fig. 24B

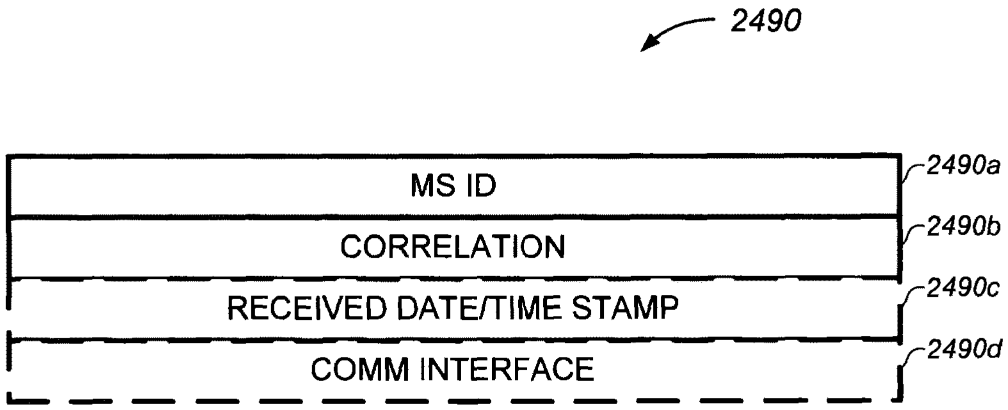

2490
MS ID
2490a
CORRELATION
2490b
RECEIVED DATE/TIME STAMP
2490c
COMM INTERFACE
2490d

Fig. 24C

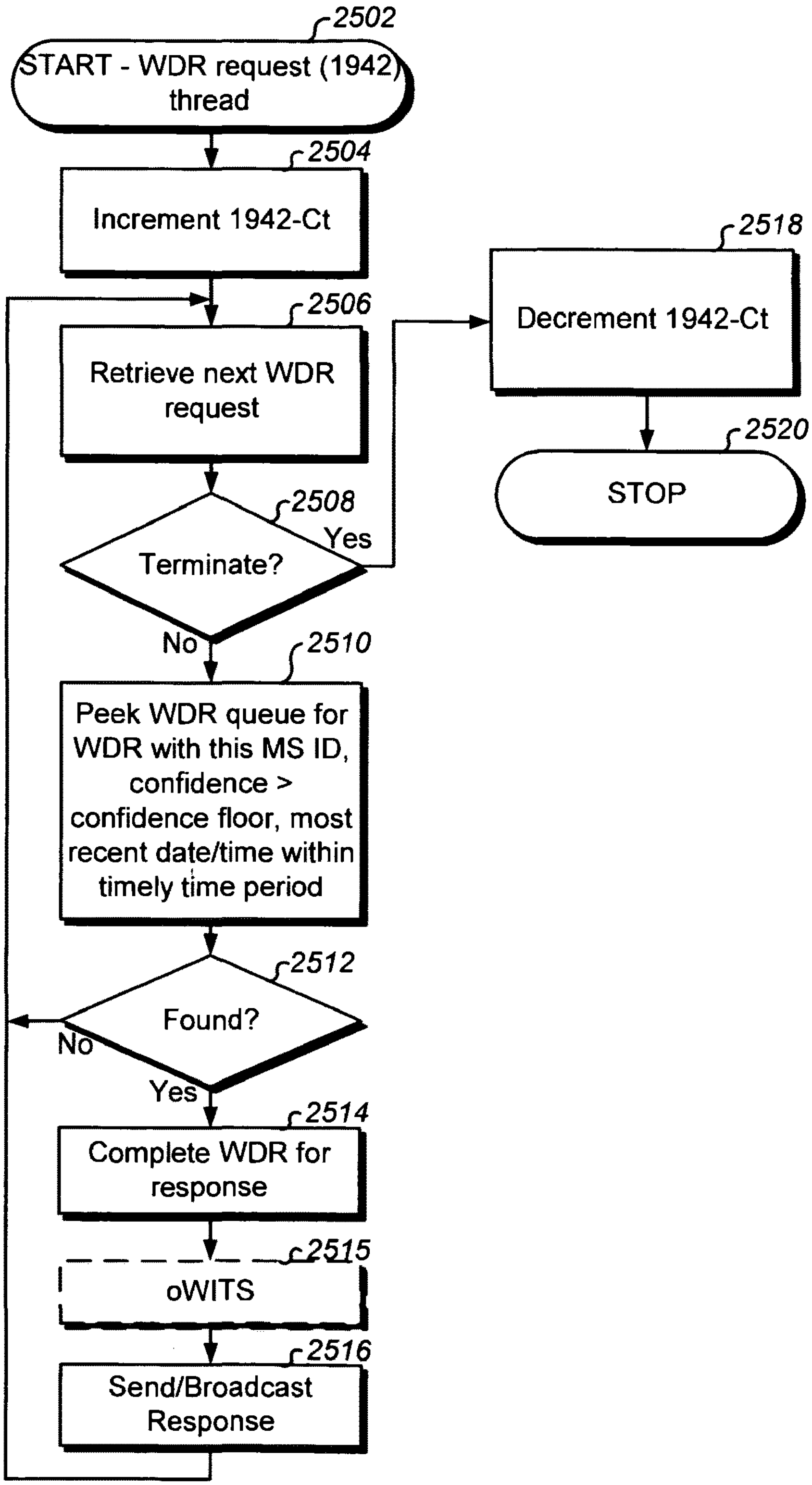

2502
START - WDR request (1942) thread
2504
Increment 1942-Ct
2506
Retrieve next WDR request
2508
Terminate?
Yes
No
2510
Peek WDR queue for WDR with this MS ID, confidence > confidence floor, most recent date/time within timely time period
2512
Found?
No
Yes
2514
Complete WDR for response
2515
oWITS
2516
Send/Broadcast Response
2518
Decrement 1942-Ct
2520
STOP

Fig. 25

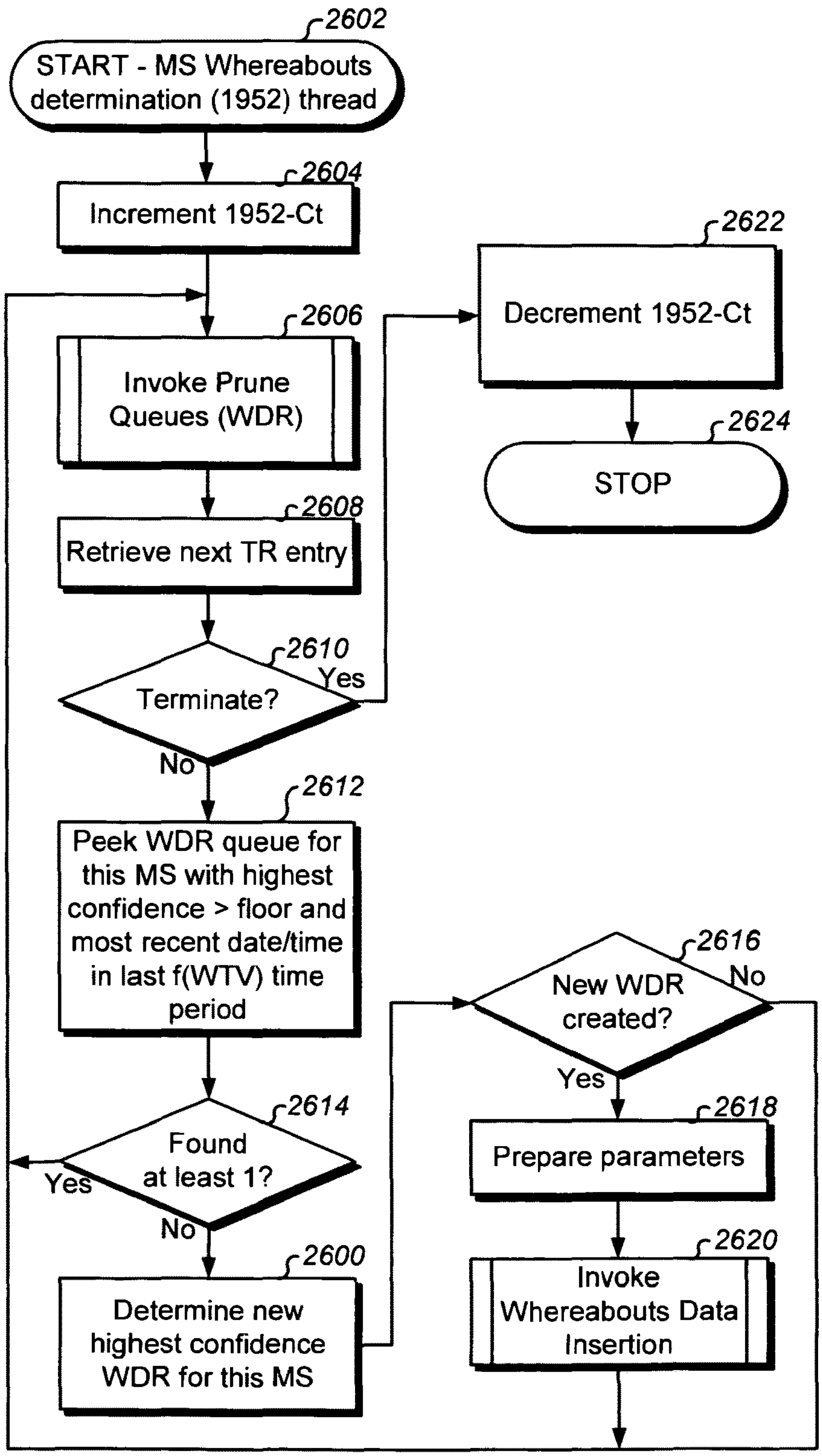

2602
START - MS Whereabouts determination (1952) thread
2604
Increment 1952-Ct
2606
Invoke Prune Queues (WDR)
2608
Retrieve next TR entry
2610
Terminate?
Yes
No
2612
Peek WDR queue for this MS with highest confidence > floor and most recent date/time in last f(WTV) time period
2614
Found at least 1?
Yes
No
2600
Determine new highest confidence WDR for this MS
2616
New WDR created?
No
Yes
2618
Prepare parameters
2620
Invoke Whereabouts Data Insertion
2622
Decrement 1952-Ct
2624
STOP

Fig. 26A

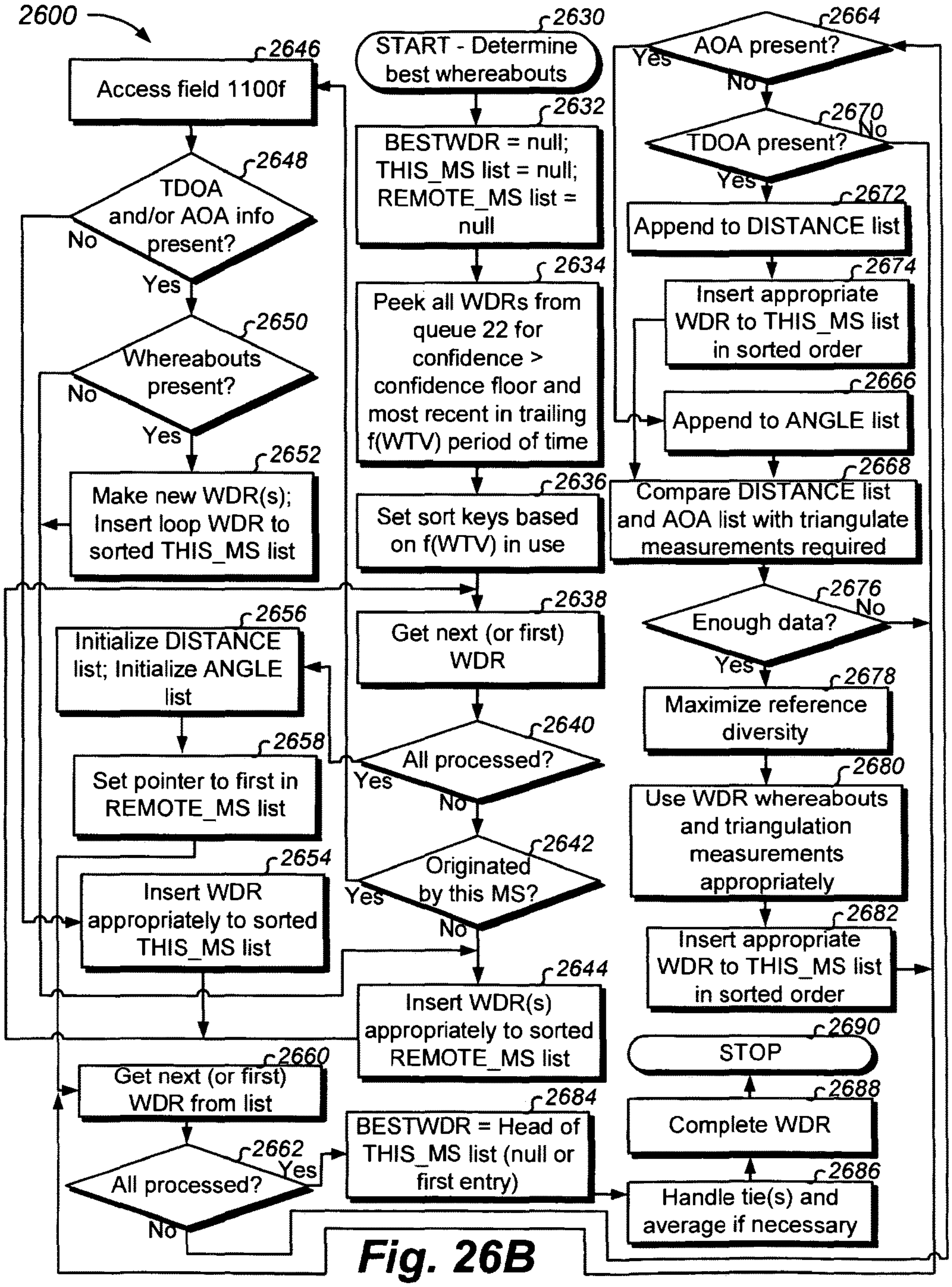

2600
2630
START - Determine best whereabouts
2632
BESTWDR = null; THIS_MS list = null; REMOTE_MS list = null
2634
Peek all WDRs from queue 22 for confidence > confidence floor and most recent in trailing f(WTV) period of time
2636
Set sort keys based on f(WTV) in use
2638
Get next (or first) WDR
2640
All processed?
Yes
No
2642
Originated by this MS?
Yes
No
2644
Insert WDR(s) appropriately to sorted REMOTE_MS list
2646
Access field 1100f
2648
TDOA and/or AOA info present?
No
Yes
2650
Whereabouts present?
No
Yes
2652
Make new WDR(s); Insert loop WDR to sorted THIS_MS list
2656
Initialize DISTANCE list; Initialize ANGLE list
2658
Set pointer to first in REMOTE_MS list
2654
Insert WDR appropriately to sorted THIS_MS list
2660
Get next (or first) WDR from list
2662
All processed?
Yes
No
2684
BESTWDR = Head of THIS_MS list (null or first entry)
2664
AOA present?
Yes
No
2670
TDOA present?
No
Yes
2672
Append to DISTANCE list
2674
Insert appropriate WDR to THIS_MS list in sorted order
2666
Append to ANGLE list
2668
Compare DISTANCE list and AOA list with triangulate measurements required
2676
Enough data?
No
Yes
2678
Maximize reference diversity
2680
Use WDR whereabouts and triangulation measurements appropriately
2682
Insert appropriate WDR to THIS_MS list in sorted order
2690
STOP
2688
Complete WDR
2686
Handle tie(s) and average if necessary

Fig. 26B

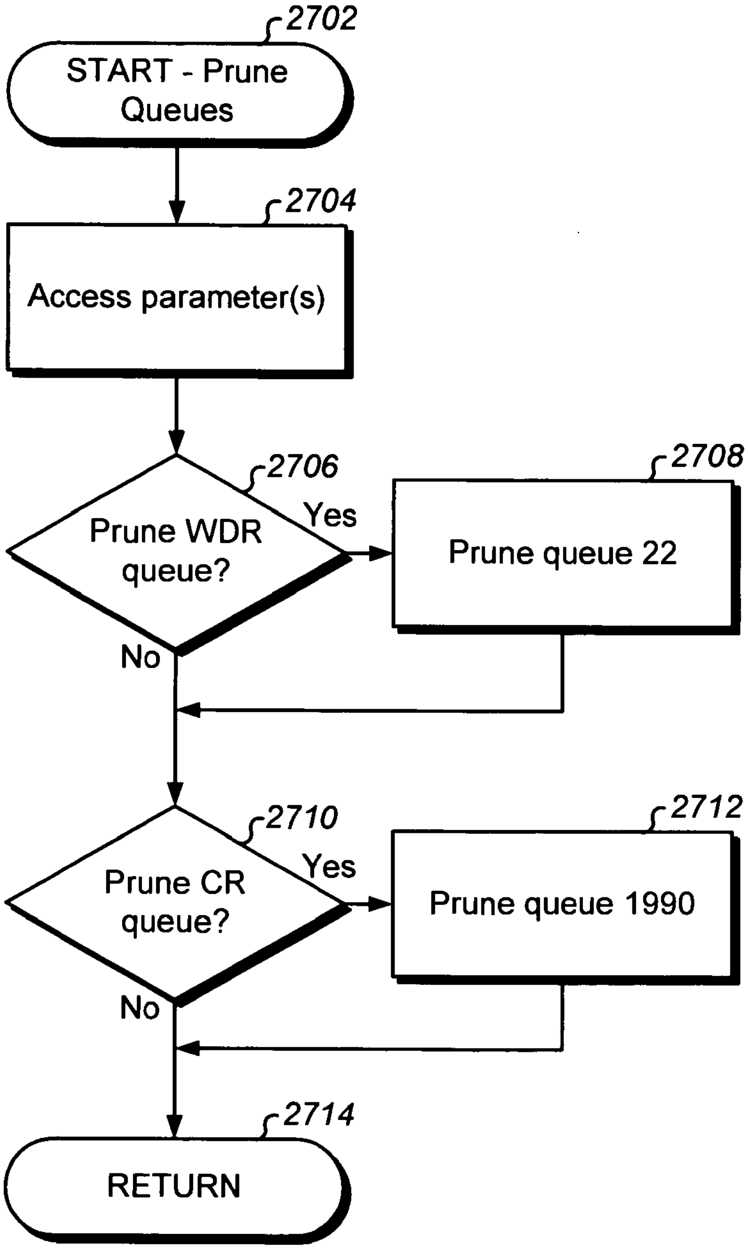

2702
START - Prune Queues
2704
Access parameter(s)
2706
Prune WDR queue?
Yes
No
2708
Prune queue 22
2710
Prune CR queue?
Yes
No
2712
Prune queue 1990
2714
RETURN

Fig. 27A

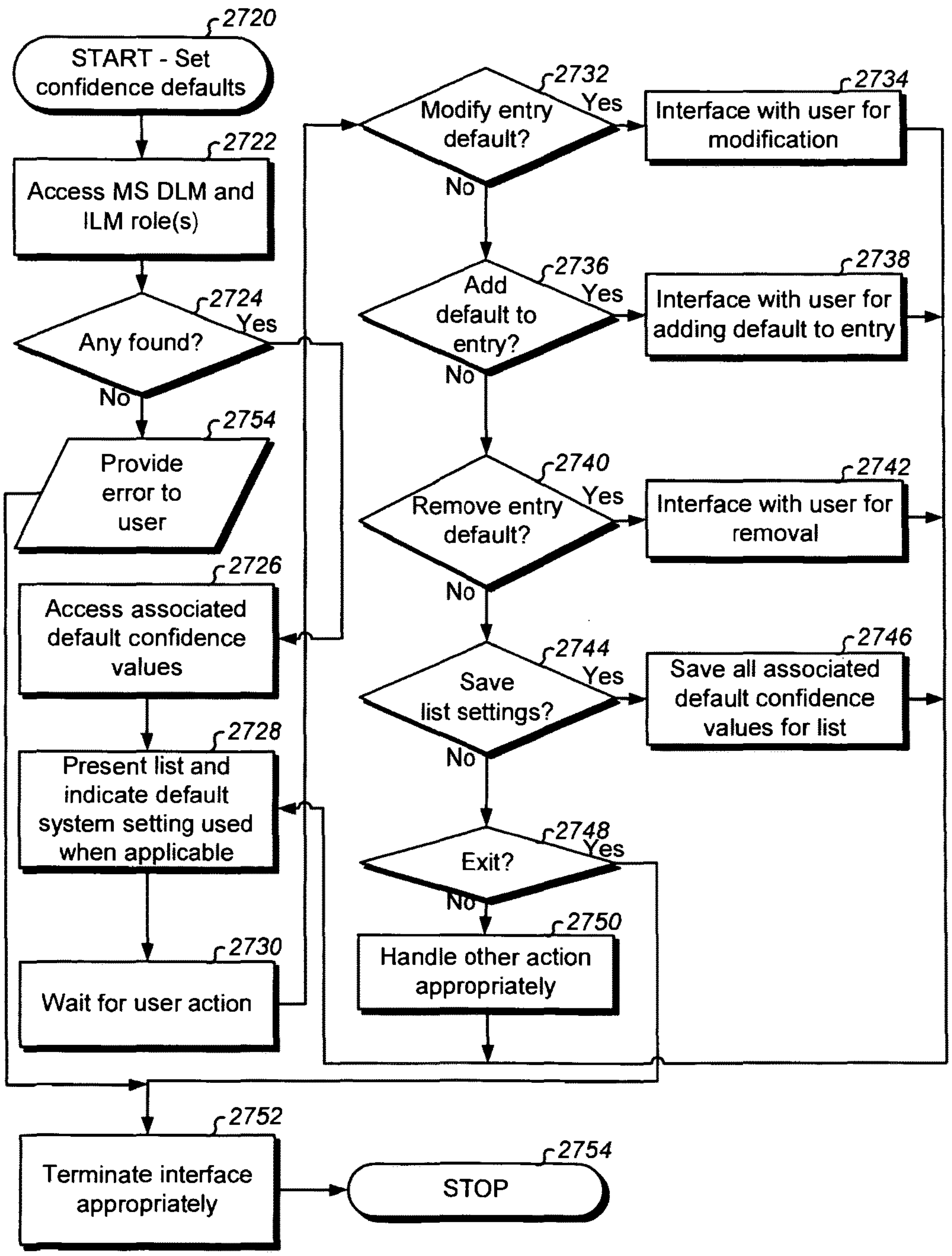

2720
START - Set confidence defaults
2722
Access MS DLM and ILM role(s)
2724
Any found?
Yes
No
2754
Provide error to user
2726
Access associated default confidence values
2728
Present list and indicate default system setting used when applicable
2730
Wait for user action
2732
Modify entry default?
Yes
No
2734
Interface with user for modification
2736
Add default to entry?
Yes
No
2738
Interface with user for adding default to entry
2740
Remove entry default?
Yes
No
2742
Interface with user for removal
2744
Save list settings?
Yes
No
2746
Save all associated default confidence values for list
2748
Exit?
Yes
No
2750
Handle other action appropriately
2752
Terminate interface appropriately
2754
STOP

Fig. 27B

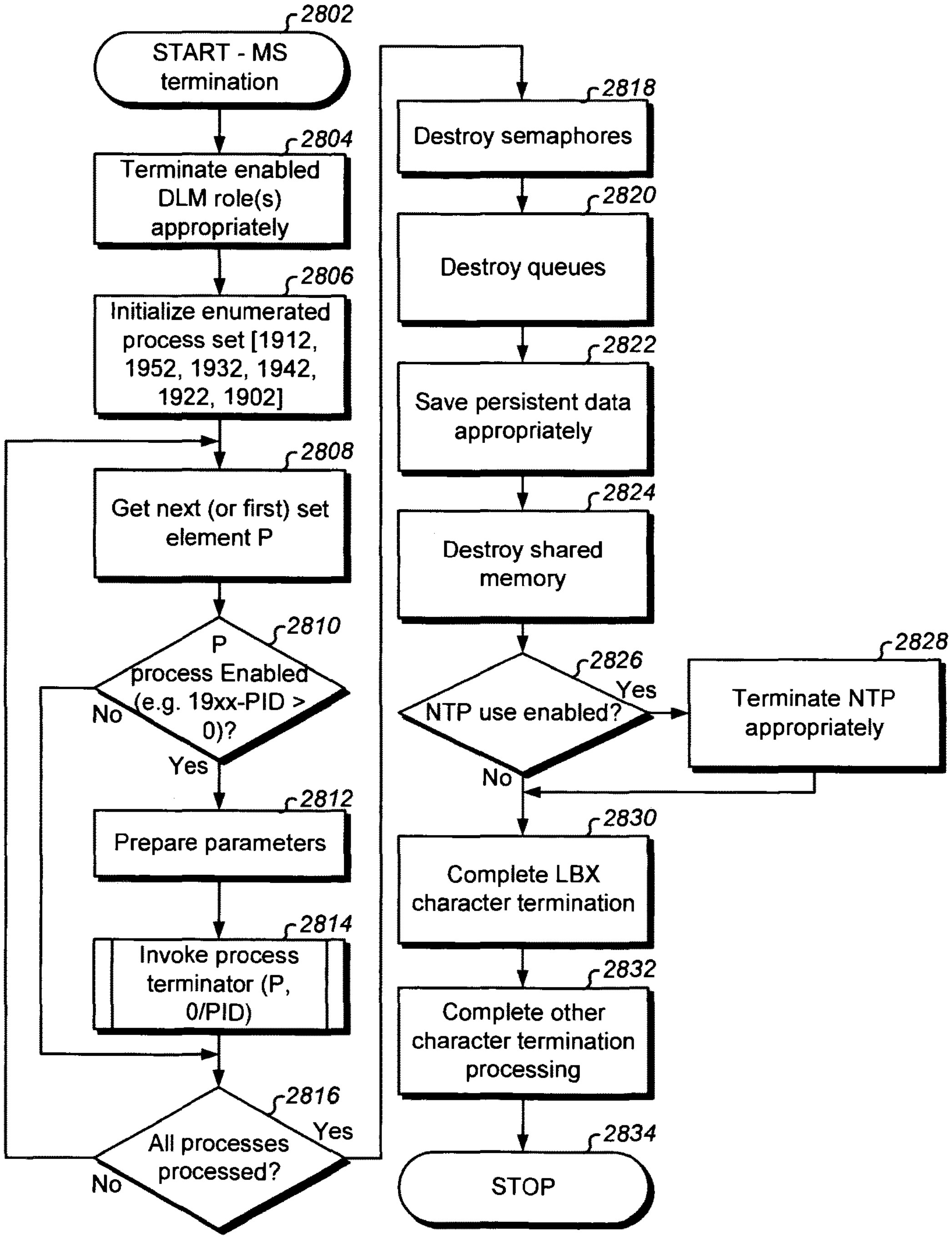

2802
START - MS termination
2804
Terminate enabled DLM role(s) appropriately
2806
Initialize enumerated process set [1912, 1952, 1932, 1942, 1922, 1902]
2808
Get next (or first) set element P
2810
P process Enabled (e.g. 19xx-PID > 0)?
No
Yes
2812
Prepare parameters
2814
Invoke process terminator (P, 0/PID)
2816
All processes processed?
Yes
No
2818
Destroy semaphores
2820
Destroy queues
2822
Save persistent data appropriately
2824
Destroy shared memory
2826
NTP use enabled?
Yes
No
2828
Terminate NTP appropriately
2830
Complete LBX character termination
2832
Complete other character termination processing
2834
STOP

Fig. 28

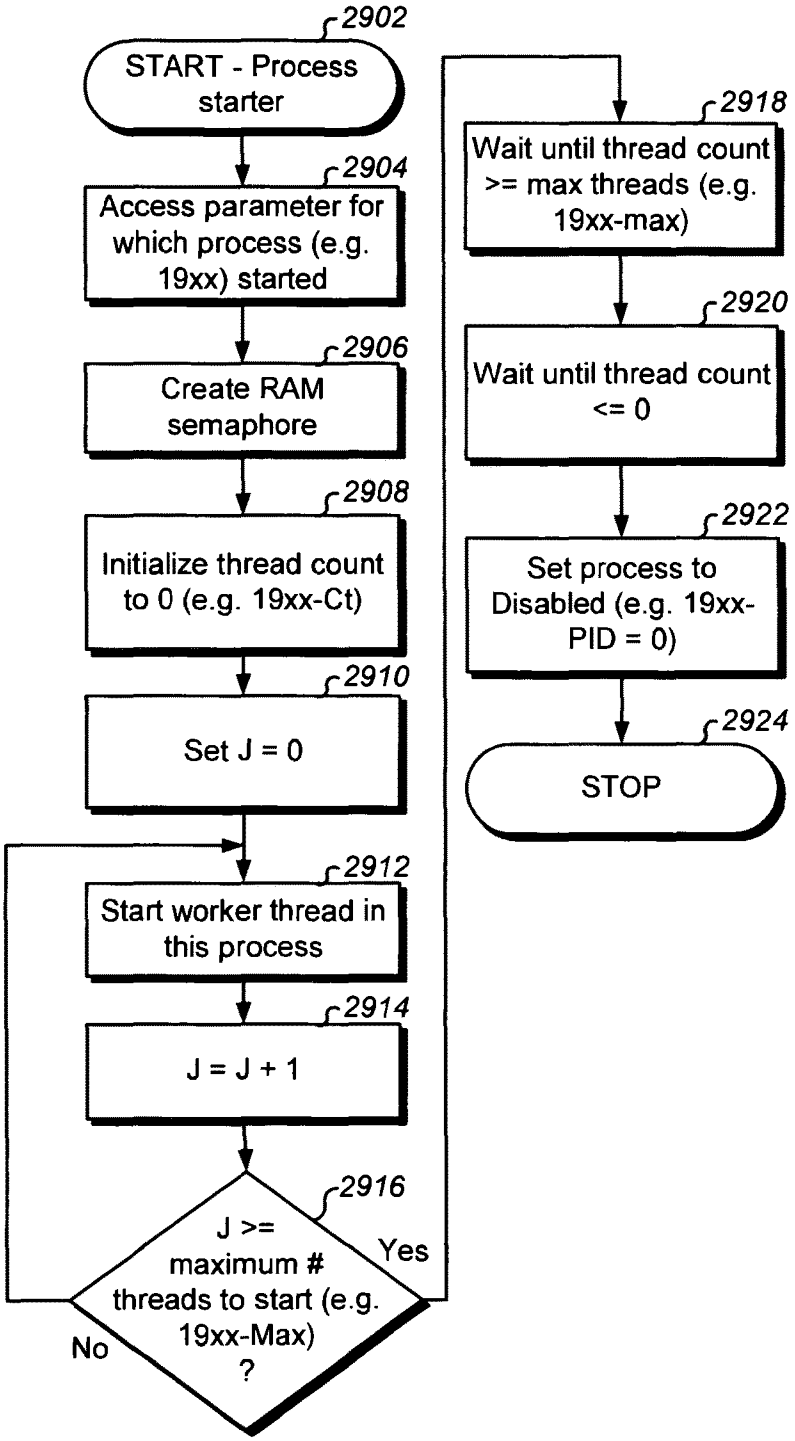

2902
START - Process starter
2904
Access parameter for which process (e.g. 19xx) started
2906
Create RAM semaphore
2908
Initialize thread count to 0 (e.g. 19xx-Ct)
2910
Set J = 0
2912
Start worker thread in this process
2914
J = J + 1
2916
J >= maximum # threads to start (e.g. 19xx-Max) ?
Yes
No
2918
Wait until thread count >= max threads (e.g. 19xx-max)
2920
Wait until thread count <= 0
2922
Set process to Disabled (e.g. 19xx-PID = 0)
2924
STOP

Fig. 29A

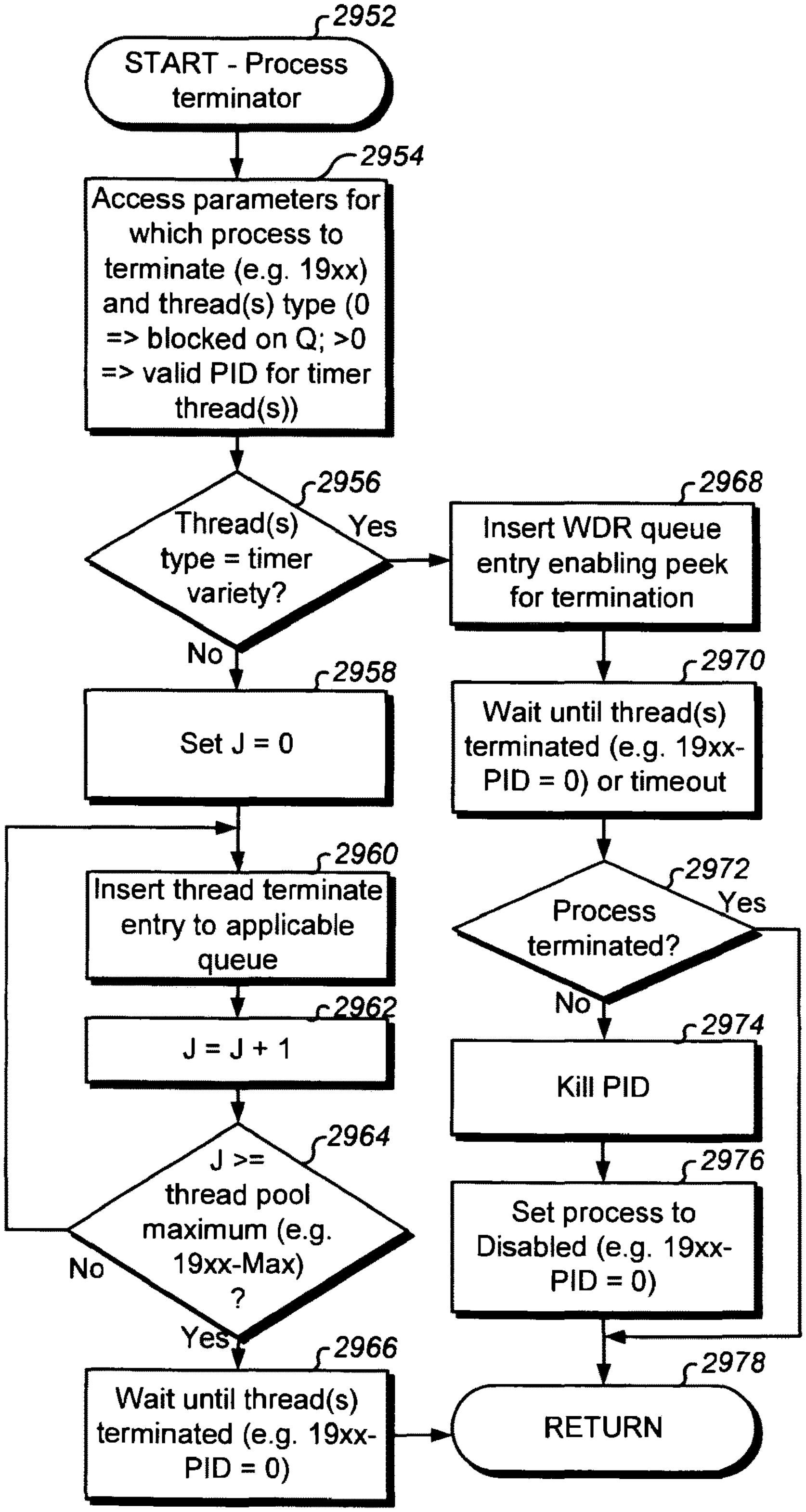

2952
START - Process terminator
2954
Access parameters for which process to terminate (e.g. 19xx) and thread(s) type (0 => blocked on Q; >0 => valid PID for timer thread(s))
2956
Thread(s) type = timer variety?
Yes
No
2968
Insert WDR queue entry enabling peek for termination
2970
Wait until thread(s) terminated (e.g. 19xx-PID = 0) or timeout
2972
Process terminated?
Yes
No
2974
Kill PID
2976
Set process to Disabled (e.g. 19xx-PID = 0)
2978
RETURN
2958
Set J = 0
2960
Insert thread terminate entry to applicable queue
2962
J = J + 1
2964
J >= thread pool maximum (e.g. 19xx-Max) ?
No
Yes
2966
Wait until thread(s) terminated (e.g. 19xx-PID = 0)

Fig. 29B

3002a

```
// Figs. 30A through 30E syntaxes (e.g. delimiters, etc) used should enforce
// appropriate unambiguous grammar parsability for Lex&Yacc, top down
// recursive parsing, XML encoding, other syntactic embodiments, applicable semantic
// representations, and any other syntactic/semantic embodiments. Figs. 30A through 30E BNF
// grammar elaborates for a corresponding interpreter, recommended syntaxes, programming
// language structures and/or objects, DB schemas, ANSI datastream encoding (e.g. X.409),
// flowchart processing blocks and locations in parent application flowcharts, and any other
// analogous implementation embodiments or subsets thereof.

// ***** Common BNF grammar (e.g. in Data 8): *****

Variables            = "null" | Variables Variable
        // Variables are placed anywhere; Can be used for referencing (a="..." b=a  c=b)

Variable             = VarType(VarName) = "null" | VarType(VarName) = ...value(s)... |
                           VarType(VarName) = [ Variables ] [ VarInstantiations ] |
                           VarType(VarName) = [ VarInstantiations ]  [ Variables ]
        // Variables scope to following & descending nesting; "value" has appropriate syntax
        // per VarType; VarName can be set to other variables (e.g. indirect tree structure)

VarInstantiations    = "null" | VarInstantiations VarInstantiate

VarInstantiate       = *VarName(Param1="x1", Param2="x2", ... ParamN="xN")  for N >= 0
        // Parameters allow optionally substituting occurrences in VarName with new values
        // prior to instantiation.

VarName              = "text string"

Description          = "null" | "text string" | VarInstantiate

History              = [ CreatorInfo ] [ ModifierInfo ] | VarInstantiations

CreatorInfo          = "null" | [ CreateDateTime ] [ CreatorID] [ CreatorIDType ]
                           [ CreatorAddr ] [ CreatorSysID ] [ CreatorSysType ]
                           [ CreatorSysAddr ] | VarInstantiations

ModifierInfo         = "null" | [ LastModifyDateTime ] [ LastModifyID ]
               [ LastModifyIDType ] [ LastModifyAddr ] [ LastModifySysID ]
               [ LastModifySysType ] [ LastModifySysAddr ] | VarInstantiations

CreateDateTime       = "date/time stamp" | VarInstantiate

CreatorID            = ID

CreatorIDType        = IDType
```

Fig. 30A

3002b

```
CreatorAddr          =  Address

CreatorSysID         = "text string" | VarInstantiate

CreatorSysType       =  "system type" | VarInstantiate   // e.g. type of MS

CreatorSysAddr       =  Address

LastModifyDateTime=  "date/time stamp" | VarInstantiate

LastModifyID         =  ID

LastModifyIDType     =  IDType

LastModifyAddr       =  Address

LastModifySysID      = "text string" | VarInstantiate

LastModifySysType  =  "system type" | VarInstantiate

LastModifySysAddr  =  Address

ID                   =  "MS ID" [ Description ] [ History ] |
        "MS Group ID" [ Description ] [ History ] | "User ID" [ Description ] [ History ] |
        "User Group ID" [ Description ] [ History ] | "logical handle" [ Description ] [ History ] |
        "physical handle" [ Description ] [ History ] | VarInstantiations

IDType               =  "MS_ID" | "MS_Group_ID" | "User_ID" | "User_Group_ID"  |
                            "logical_handle" | "physical_handle" | VarInstantiate

Address              = "ip address" | "SNA address" | "Postal address" |
                      "point" | "logical address" | "physical address" | "situational location" |
                      "2 dimensional area" | "3 dimensional area" | VarInstantiate

TimeSpec             =  "Xdate/time stamp" | "Xdate/time period" | VarInstantiate

VarType              =  Description | History | ID | IDType | CreatorInfo | ModifierInfo |
        CreateDateTime | CreatorID | CreatorIDType | CreatorAddr | CreatorSysID |
        CreatorSysType | CreatorSysAddr | LastModifyDateTime | LastModifyID |
        LastModifyIDType | LastModifyAddr | LastModifySysID | LastModifySysType |
        LastModifySysAddr | Address | "Xdate/time stamp" |  "Xdate/time period" | "text string" |
        "system type" | TimeSpec | "MS ID" | "MS Group ID" | "User ID" | "User Group ID" |
        "logical handle" | physical handle" | "...Address elaborations..." |
        "...IDType elaborations..." |  Variable  // | VarInstantiate here as well (but elaborates)
```

Fig. 30B

3034

```
// ***** BNF grammar for Permissions 10: *****

PermissionBody     = "null" | [ Variables ] [ Permissions ]
        // [ Variables ] placed anywhere (not shown in constructs below to enhance readability)

Permissions        = "null" | Permissions Permission | VarInstantiations

Permission         = Grantor Grantee [ Grants ] [ TimeSpec ] [ Description ] [ History ] |
                          VarInstantiations
        // No Grants implies granting all permissions; This embodiment ensures non-null
        // Grantor and Grantee, but "null" could be used (e.g. for placeholder entries).

Grantor            = ID [ IDType ] | VarInstantiations
        // ID defaults (e.g. MS ID) when IDType not present

Grantee            = ID [ IDType ] | VarInstantiations

Grants             = "null" | Grants Grant | Privileges | VarInstantiations

Grant              = "grant name" AND (Privileges [ TimeSpec ] [ Description ] [ History ] |
                          Grants  [ TimeSpec ] [ Description ] [ History ] |
                          VarInstantiations)

Privileges         = "null" | Privileges Privilege | VarInstantiations

Privilege          = "atomic privilege for assignment" [ MSRelevance ]
                              [ TimeSpec ] [ Description ] [ History ] | VarInstantiations

MSRelevance        = "MS relevance descriptor"

Groups             = "null" | Groups Group | VarInstantiations

Group              = "group name" AND (IDs [ Description ] [ History ] |
                          Groups  [ Description ] [ History ] |
                          VarInstantiations)

IDs                = "null" | IDs ID [ IDType ] |  VarInstantiations

VarType            = *VarType | Permissions | Permission | Grantor | Grantee | Grants |
                          Grant | Privileges | Privilege | MSRelevance | Groups | Group |
                          IDs
```

Fig. 30C

3068a

```
// ***** BNF grammar for Charters 12: *****

CharterBody         = "null" | [ Variables ] [ Charters ]
        // [ Variables ] placed anywhere (not shown in constructs below to enhance readability)

Charters            = "null" | Charters Charter | VarInstantiations

Charter             = Grantee Grantor Expression Actions [ TimeSpec ] [ Description ]
                             [ History ] | VarInstantiations

Expression          = Conditions [ TimeSpec ] | VarInstantiations
        // This embodiment ensures at least one condition to a Charter, but "null" could be
        // used (e.g. for placeholder entries).

Conditions          = Condition | Conditions CondOp Condition] | VarInstantiations

CondOp              = "and" | "or" | VarInstantiations

Condition           = Term Op Term [ TimeSpec ] [ Description ] [ History ] |
        Value [ TimeSpec ] [ Description ] [ History ] |
        Invocation [ TimeSpec ] [ Description ] [ History ] | VarInstantiations
        // Another embodiment allows unary operators (e.g. "not"), for example for boolean
        // WDR fields (e.g. Applications field(s)). Current boolean tests for "True" or "False",
        // or non-zero = "True" and zero = "False". Value & Invocation result in a boolean.

Term                = WDRTerm  [ TimeSpec ] [ Description ] [ History ] |
                      AppTerm  [ TimeSpec ] [ Description ] [ History ] |
                      Value  [ TimeSpec ] [ Description ] [ History ] |
                      Invocation  [ TimeSpec ] [ Description ] [ History ] |
                      PointSet  [ TimeSpec ] [ Description ] [ History ]  |
                       VarInstantiate

WDRTerm             = "Any WDR 1100 field, or any subset thereof" [ Description ]
                             [ History ] | VarInstantiate

AppTerm             = "Any Application data field, or any subset thereof" [ Description ]
                             [ History ] | VarInstantiate

Value               = Data | "number" | "text string" | "value" | "True" | "False" |
                             "atomic term" | "map term" | ID [ IDType ] | "null" | VarInstantiate

PointSet            = [2D | 3D]  [Geo | Cartesian | Polar]
                             "text string" [ Description ] [ History ]  |
                              "numeric(s)" [ Description ] [ History ] |
                              Data [ Description ] [ History ]  | VarInstantiate
```

Fig. 30D

3068b

```
Data               = "typed memory pointer" | "typed memory value" | "typed file path" |
                          "typed file path and offset" | "typed DB qualifier" | VarInstantiate
        // i.e. pointer or value from stack, globals, shared memory, file data location, DB
        // pointer, DB value, or any other data.

Invocation         = "DLL interface(optional params...)" |
                          "Linked interface(optional params...)" |
                           "executable path(optional params...)" | VarInstantiate
        // Invocation can return any value of any type, except will be converted to a boolean
        // when used as a Term (0 = False, else = True). Best to return boolean when Term use.

Op                 = [ "atomic not operator" ] "atomic operator" | ProfileMatch |
                          VarInstantiate

ProfileMatch       = "atomic profile match operator" | VarInstantiate

Actions            = "null" | Actions Action

Action             =  [ Host ] Command Operand [Parameters]
                          [ TimeSpec ] [ Description ] [ History ] | VarInstantiations

Host               = "null" | ID [IDType] | VarInstantiations

Command            =  "atomic command" | VarInstantiations
        // Command may map to translation member entry of natural language map

Operand            =  "atomic operand" | VarInstantiations
        // Some embodiments have no need for an operand in this grammar (e.g. command file
        // reference, DLL call, self contained command, invocation callout, etc).

Parameters         =  "null" | Parameters Parameter | VarInstantiations

Parameter          =  WDRTerm  [ Description ] [ History ] |
                          AppTerm  [ Description ] [ History ] |
                          Value [ Description ] [ History ] |
                          Invocation [ Description ] [ History ] |
                          ID [ IDType ] [ Description ] [ History ] |
                          VarInstantiate [ Description ] [ History ]

VarType            =  *VarType | Charters | Charter | Expression | Conditions | Condition |
                   CondOp | WDRTerm | Term | Value | PointSet | Data | Invocation | Op |
                   Actions |  ProfileMatch | Action | Command | Operand | Parameters |
                   Parameter | Host
```

Fig. 30E

Operand ↓	Command										
	101	103	105	119	107	109	111	113	115	117	. . .
201	#, sender, msg/subj, attribs, recip(s)	#, sender, msg/subj, attribs, recip(s)	#	#	#, system(s)	#, system(s)	#, ack, source, system(s)	#, ack, system(s)	#, ack, source, system(s)	#, system(s)	
203	link, sender, msg/subj, attribs, recip(s)	link, sender, msg/subj, attribs, recip(s)	link, params	link, params	link, params, system(s)	link, params, system(s)	link, ack, source, system(s)	link, ack, system(s)	link, ack, source, system(s)	link, params, system(s)	
205	body, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	email, system(s)	body, sender, msg/subj, attribs, recip(s)	email, ack, source, system(s)	email, ack, system(s)	email, ack, source, system(s)	email, system(s)	
207	msg, sender, msg/subj, attribs, recip(s)	msg, sender, msg/subj, attribs, recip(s)	msg, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	msg, system(s)	body, sender, msg/subj, attribs, recip(s)	msg, ack, source, system(s)	msg, ack, system(s)	msg, ack, source, system(s)	msg, system(s)	
209	body, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	email, system(s)	body, sender, msg/subj, attribs, recip(s)	email, ack, source, system(s)	email, ack, system(s)	email, ack, source, system(s)	email, system(s)	
211	msg, sender, msg/subj, attribs, recip(s)	msg, sender, msg/subj, attribs, recip(s)	msg, sender, msg/subj, attribs, recip(s)	msg, sender, msg/subj, attribs, recip(s)	msg, system(s)	body, sender, msg/subj, attribs, recip(s)	msg, ack, source, system(s)	msg, ack, system(s)	msg, ack, source, system(s)	msg, system(s)	
213	indicator, sender, msg/subj, attribs, recip(s)	indicator, sender, msg/subj, attribs, recip(s)	indicator, sender, msg/subj, attribs, recip(s)	indicator, sender, msg/subj, attribs, recip(s)	indicator, system(s)	indicator, system(s)	indicator, ack, source, system(s)	indicator, ack, system(s)	indicator, ack, source, system(s)	indicator, system(s)	

Fig. 31A

Operand ↓	Command										
	101	103	105	119	107	109	111	113	115	117	. . .
215	app, sender, msg/subj, attribs, recip(s)	app, sender, msg/subj, attribs, recip(s)	app, params	app, params	app, params, system(s)	app, params, system(s)	app, params, ack, source, system(s)	app, params, ack, system(s)	app, params, ack, source, system(s)	app, params, system(s)	
217	doc, sender, msg/subj, attribs, recip(s)	doc, sender, msg/subj, attribs, recip(s)	doc	doc	doc, system(s)	doc, system(s)	doc, ack, source, system(s)	doc, ack, system(s)	doc, ack, source, system(s)	doc, system(s)	
219	path, sender, msg/subj, attribs, recip(s)	path, sender, msg/subj, attribs, recip(s)	path	path	path, system(s)	path, system(s)	path, ack, source, system(s)	path, ack, system(s)	path, ack, source, system(s)	path, system(s)	
221	content, sender, msg/subj, attribs, recip(s)	content, sender, msg/subj, attribs, recip(s)	content	content	content, system(s)	content, system(s)	content, ack, source, system(s)	content, ack, system(s)	content, ack, source, system(s)	content, system(s)	
223	DB-obj, sender, msg/subj, attribs, recip(s)	DB-obj, query, sender, msg/subj, attribs, recip(s)	DB-obj	DB-obj, query	DB-obj, system(s)	DB-obj, query, system(s)	DB-obj, ack, source, system(s)	DB-obj, ack, system(s)	DB-obj, ack, source, system(s)	DB-obj, query, system(s)	
225	data, sender, msg/subj, attribs, recip(s)	data, value, sender, msg/subj, attribs, recip(s)	data, value	data, value	data, system(s)	data, value, system(s)	data, ack, source, system(s)	data, ack, system(s)	data, ack, source, system(s)	data, value, system(s)	

Fig. 31B

Operand ↓	*Command*										
	101	**103**	**105**	**119**	**107**	**109**	**111**	**113**	**115**	**117**	. . .
227	sem, sender, msg/subj, attribs, recip(s)	sem, cmd, sender, msg/subj, attribs, recip(s)	sem, cmd	sem, cmd	sem, system(s)	sem, cmd, system(s)	sem, ack, source, system(s)	sem, ack, system(s)	sem, ack, source, system(s)	sem, cmd, system(s)	
229	path, sender, msg/subj, attribs, recip(s)	path, sender, msg/subj, attribs, recip(s)	path	path	path, system(s)	path, system(s)	path, ack, source, system(s)	path, ack, system(s)	path, ack, source, system(s)	path, system(s)	
231	app, macro, sender, msg/subj, attribs, recip(s)	app, macro, sender, msg/subj, attribs, recip(s)	app, macro	app, macro	app, macro, system(s)	app, macro, system(s)	app, params, ack, source, system(s)	app, params, ack, system(s)	app, params, ack, source, system(s)	app, macro, system(s)	
233	"<alt> <prtscr>", sender, msg/subj, attribs, recip(s)	"<alt> <prtscr>", sender, msg/subj, attribs, recip(s)	"<alt> <prtscr>"	"<alt> <prtscr>"	objtxt, system(s)	cmds, system(s)	"<alt> <prtscr>", ack, source, system(s)	objtxt, ack, system(s)	"<alt> <prtscr>", ack, source, system(s)	"<alt> <prtscr>", system(s)	
235	macro, sender, msg/subj, attribs, recip(s)	macro, sender, msg/subj, attribs, recip(s)	macro	macro	macro, system(s)	macro, system(s)	macro, ack, system(s)	app, params, ack, system(s)	macro, ack, system(s)	macro, system(s)	
237	iodev, input, sender, msg/subj, attribs, recip(s)	iodev, input, sender, msg/subj, attribs, recip(s)	iodev, input	iodev, input	iodev, input, system(s)	iodev, input, system(s)	iodev, input, ack, system(s)	iodev, ack, system(s)	iodev, input, ack, system(s)	iodev, input, system(s)	

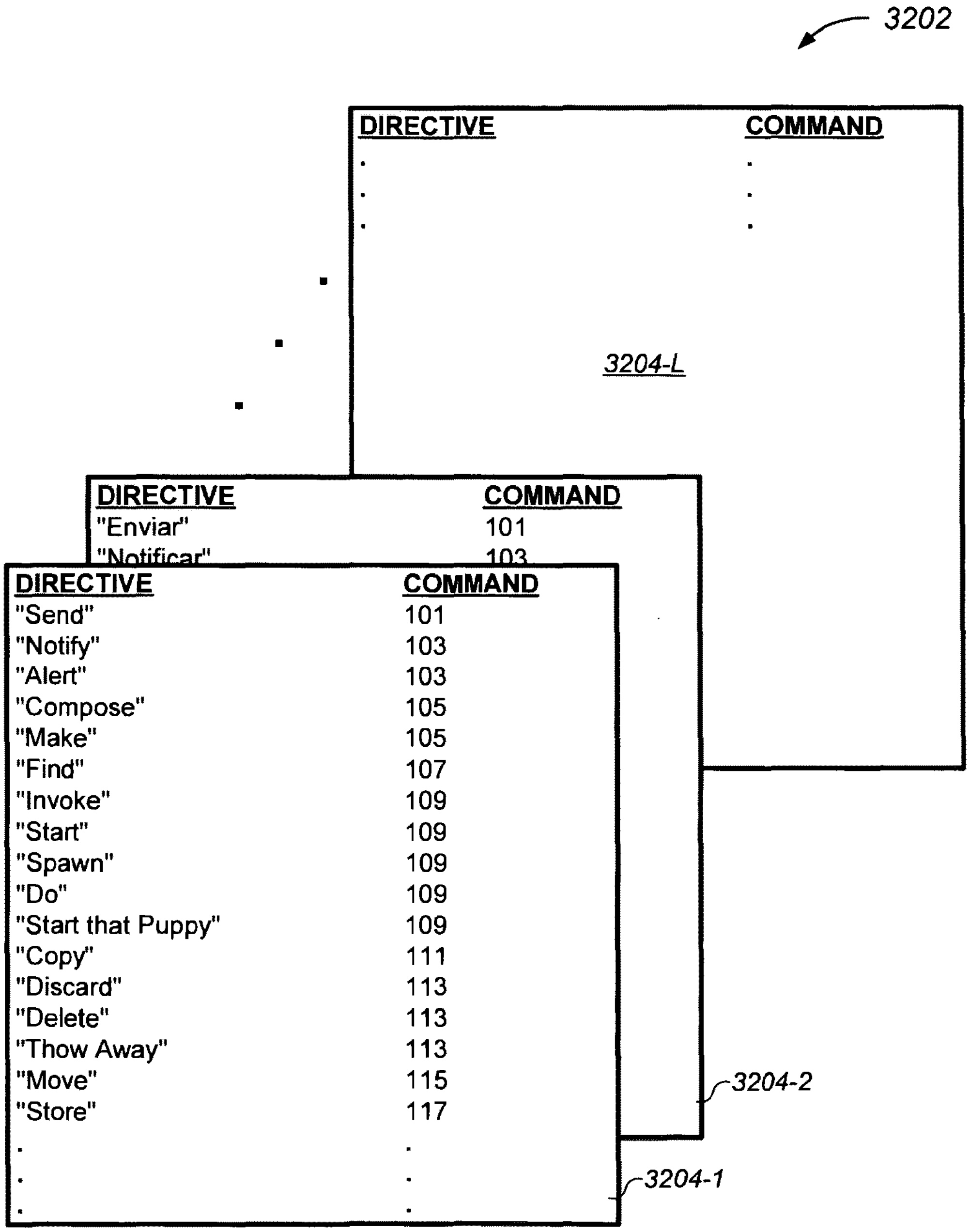
Fig. 31C

Operand ↓	Command										
	101	103	105	119	107	109	111	113	115	117	. . .
239	iodev, output, sender, subj, attribs, recip(s)	iodev, output, sender, subj, attribs, recip(s)	iodev, output	iodev, output	iodev, output, system(s)	iodev, output, system(s)	iodev, output, ack, system(s)	iodev, ack, system(s)	iodev, output, ack, system(s)	iodev, output, system(s)	
241	alert, sender, msg/subj, attribs, recip(s)	alert, sender, msg/subj, attribs, recip(s)	alert	alert	alert, system(s)	alert, system(s)	alert, ack, source, system(s)	alert, ack, system(s)	alert, ack, source, system(s)	alert, system(s)	
243	pid, signal, sender, msg/subj, attribs, recip(s)	pid, signal, sender, msg/subj, attribs, recip(s)	pid, signal	pid, signal	prname, system(s)	pid, signal, system(s)	prname, ack, source, system(s)	prname, ack, system(s)	prname, ack, source, system(s)	prname, signal system(s)	
245	container, sender, msg/subj, attribs, recip(s)	container, sender, msg/subj, attribs, recip(s)	container	container	container, system(s)	container, system(s)	container, ack, source, system(s)	container, ack, system(s)	container, ack, source, system(s)	container, system(s)	
247	progobj, data, sender, msg/subj, attribs, recip(s)	progobj, data, sender, msg/subj, attribs, recip(s)	progobj, data	progobj, data, sender, msg/subj, attribs, recip(s)	progobj, data, system(s)	progobj, data, system(s)	progobj, ack, source, system(s)	progobj, ack, system(s)	progobj, ack, source, system(s)	progobj, data, system(s)	
249	cursor, sender, msg/subj, attribs, recip(s)	cursor, sender, msg/subj, attribs, recip(s)	cursor	cursor, sender, msg/subj, attribs, recip(s)	cursor, system(s)	cursor, attribs, system(s)	ack, source, system(s)	ack, system(s)	ack, source, system(s)	cursor, attribs, system(s)	

Fig. 31D

Operand ↓	*Command*										
	101	**103**	**105**	**119**	**107**	**109**	**111**	**113**	**115**	**117**	. . .
251	calobj, sender, msg/subj, attribs, recip(s)	calobj, sender, msg/subj, attribs, recip(s)	calobj, sender, msg/subj, attribs, recip(s)	calobj, sender, msg/subj, attribs, recip(s)	calobj, system(s)	calobj, attribs, system(s)	calobj, ack, source, system(s)	calobj, ack, system(s)	calobj, ack, source, system(s)	calobj, attribs, system(s)	
253	ABobj, sender, msg/subj, attribs, recip(s)	ABobj, sender, msg/subj, attribs, recip(s)	ABobj, sender, msg/subj, attribs, recip(s)	ABobj, sender, msg/subj, attribs, recip(s)	ABobj, system(s)	ABobj, attribs, system(s)	ABobj, ack, source, system(s)	ABobj, ack, system(s)	ABobj, ack, source, system(s)	ABobj, attribs, system(s)	
...											

Fig. 31E

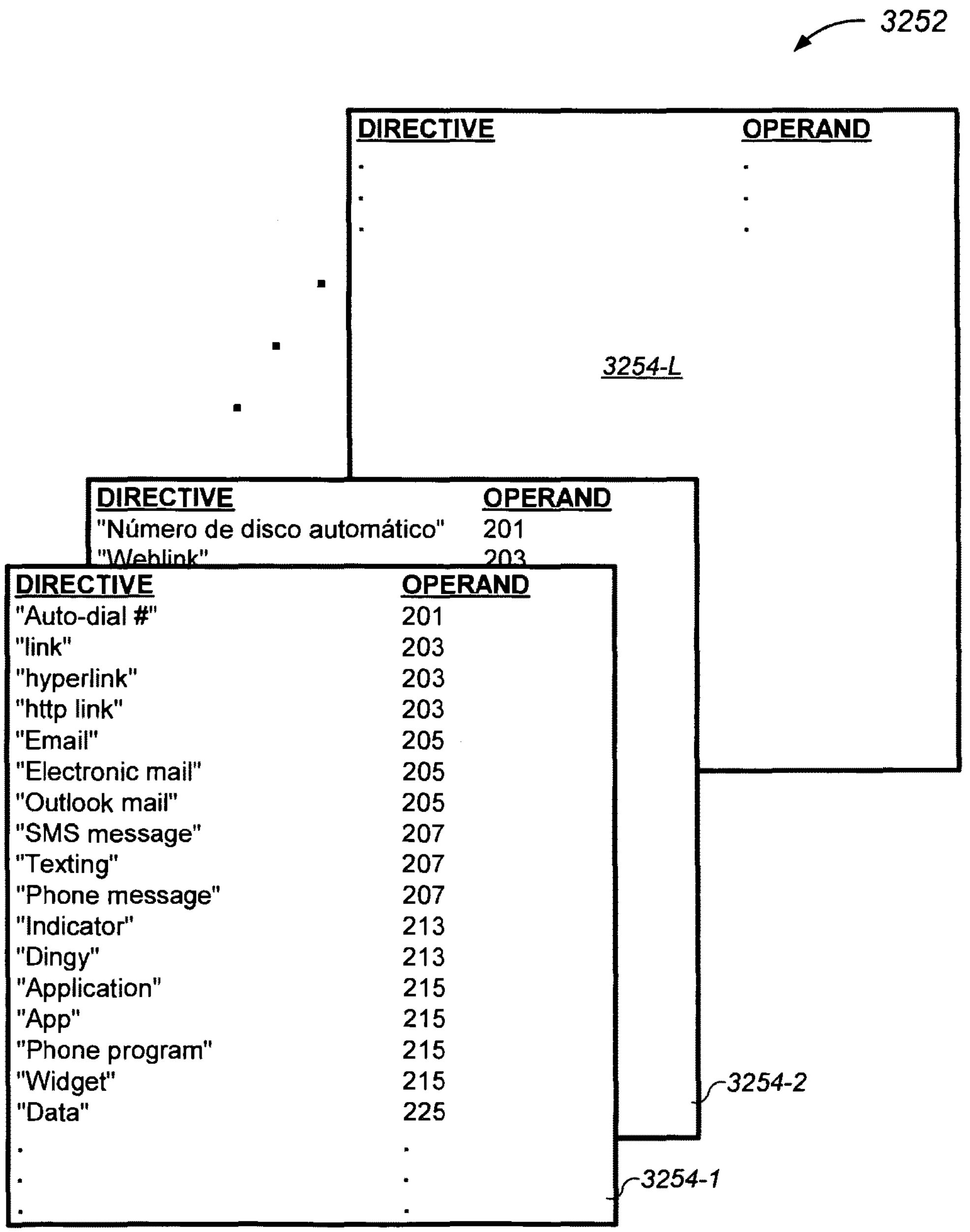


DIRECTIVE	COMMAND
.	.
.	.
.	.

DIRECTIVE	COMMAND
"Enviar"	101
"Notificar"	103

DIRECTIVE	COMMAND
"Send"	101
"Notify"	103
"Alert"	103
"Compose"	105
"Make"	105
"Find"	107
"Invoke"	109
"Start"	109
"Spawn"	109
"Do"	109
"Start that Puppy"	109
"Copy"	111
"Discard"	113
"Delete"	113
"Thow Away"	113
"Move"	115
"Store"	117
.	.
.	.
.	.

Fig. 32A



DIRECTIVE	OPERAND
.	.
.	.
.	.

DIRECTIVE	OPERAND
"Número de disco automático"	201
"Weblink"	203

DIRECTIVE	OPERAND
"Auto-dial #"	201
"link"	203
"hyperlink"	203
"http link"	203
"Email"	205
"Electronic mail"	205
"Outlook mail"	205
"SMS message"	207
"Texting"	207
"Phone message"	207
"Indicator"	213
"Dingy"	213
"Application"	215
"App"	215
"Phone program"	215
"Widget"	215
"Data"	225
.	.
.	.
.	.

Fig. 32B

Token	Length	Value
Variables[1]	L	complex ([Variable] ... [Variable]).
Variable	L	complex (First 2 bytes = VarType; VarName; value(s)); this can be present in any complex datastream for scope within current complex datastream thereafter and all descending constructs to it.
VarInstantiations[1]	L	complex ([VarInstatiate] ... [VarInstantiate]).
VarInstantiate	L	instantiation variable name and optional parameters; this can be subordinate to any other construct (e.g. {Description,8,{VarInstantiate,2,x}} where x is variable of string type (e.g. x = "Very long description text here"). Note the savings in TLV datastream size by using variables defined in 1 place for multiple subsequent instantiations thereafter).
VarName	L	text string for variable name.
Description	L	text string for description.
History	L	complex ([CreatorInfo] [ModifierInfo])
CreatorInfo	L	complex ([CreateDateTime] [CreatorID] [CreatorIDType] [CreatorAddr] [CreatorSysID] [CreatorSysType] [CreatorSysAddr]).
ModifierInfo	L	complex ([LastModifyDateTime] [LastModifyID] [LastModifyIDType] [LastModifyAddr] [LastModifySysID] [LastModifySysType] [LastModifySysAddr]).
CreateDateTime	L	date/time stamp (i.e. YYYYMMDDHHMMSS.12..J).
CreatorID	L	complex (ID).
CreatorIDType	1	Atomic element of IDType.
CreatorAddr	L	complex (Address).
CreatorSysID	L	Text string for system ID.
CreatorSysType	L	Text string for system type.
CreatorSysAddr	L	complex (Address).
LastModifyDateTime	L	date/time stamp.
LastModifyID	L	complex (ID).
LastModifyIDType	L	Atomic element of IDType.
LastModifyAddr	L	complex (Address).
LastModifySysID	L	Text string for system ID.
LastModifySysType	L	Text string for system type.
LastModifySysAddr	L	complex (Address).
ID	L	complex (first 2 bytes = length of the identifier; followed by identifier;[Description] [History]).
IDType	1	Atomic element of IDType.
Address	L	First 2 bytes = address type; next L-2 bytes = address info.
TimeSpec	L	First byte = spec type (stamp, period); Next bytes = the time spec(s) (e.g. preferably in syntax described).

Fig. 33A

Token	Length	Value
PermissionBody	L	complex ([Variables] [Permissions]).
Permissions[1]	L	complex ([Permission] ... [Permission]).
Permission	L	complex (Grantor Grantee [Grants] [TimeSpec] [Description] [History]).
Grantor	L	complex (ID [IDType]).
Grantee	L	complex (ID [IDType]).
Grants[1]	L	complex ([Grant] … [Grant] | [Privileges]).
Grant	L	First 2 bytes = length of Grant name; following bytes = grant name string; then complex: (Privileges [TimeSpec] [Description] [History] | Grants [TimeSpec] [Description] [History]).
Privileges[1]	L	complex ([Privilege] ... [Privilege]).
Privilege	L	first 4 bytes = unsigned integer for atomic privilege assigned; following bytes (L-4) are complex: ([MSRelevance] [TimeSpec] [Description] [History])
MSRelevance	8	64 bits for up to 64 different MS types of different capabilities.
Groups[1]	L	complex ([Group] … [Group]).
Group	L	First 2 bytes = length of Group name; following bytes = group name string; then complex: (IDs [Description] [History] | Groups [Description] [History]).
IDs[1]	L	complex (ID [IDType] ... ID [IDType]).

Fig. 33B

Token	Length	Value
CharterBody	L	complex ([Variables] [Charters]).
Charters[1]	L	complex ([Charter] ... [Charter]).
Charter	L	complex (Grantee Grantor Expression Actions [TimeSpec] [Description] [History]).
Expression	L	complex (Conditions [TimeSpec]).
Conditions	L	complex (Condition | Conditions CondOp Condition]).
CondOp	1	"&" or "|".
Condition	L	complex (Term Op Term [TimeSpec] [Description] [History] | Value [TimeSpec] [Description] [History] | Invocation [TimeSpec] [Description] [History]).
Term	L	complex (WDRTerm [TimeSpec] [Description] [History] | AppTerm [TimeSpec] [Description] [History] | Value [TimeSpec] [Description] [History] | Invocation [TimeSpec] [Description] [History]).
WDRTerm	L	first 2 bytes for WDR 1100 field/subfield length; following bytes are __name syntactical reference; then, if any, is complex = [Description] [History]).
AppTerm	L	first 2 bytes for Application field length; following bytes are Prefix_name syntactical reference; then, if any, is complex = [Description] [History]).
Value	L	first byte indicates Value type; following bytes (L-1), if any, is the "number", "text string", "value", "Boolean", "null", "atomic term", "map term" or complex = (Data | ID [IDType]).
PointSet	L	first byte indicates dimension; second byte indicates type; third byte indicates format; next 2 bytes is the point set information length (LEN); following bytes is the PointSet information (may be complex for Data), following bytes (L-5-LEN), if any, is complex = [Description] [History]).
Data	L	first byte = atomic element data type; L-1 following bytes are the data syntactical reference.
Invocation	L	first byte = atomic element data type; L-1 following bytes = atomic element invocation with optional parameters.
Op	2	the operator reference (not clarifier simply provides unique operator (e.g. = and != are two operators; ProfileMatch here too). Numeric values used instead of characters here.

Fig. 33C-1

Token	Length	Value
Actions[1]	L	complex ([Action] ... [Action]).
Action	L	complex ([Host] Command Operand [Parameters] [TimeSpec] [Description] [History]).
Host	L	complex (ID [IDType]).
Command	2	the command reference.
Operand	2	the operand reference.
Parameters[1]	L	complex ([Parameter] ... [Parameter]).
Parameter	L	complex (WDRTerm [Description] [History] | AppTerm [Description] [History] | Value [Description] [History] | Invocation [Description] [History] ID [IDType] [Description] [History]).

Fig. 33C-2

```
//********* Grammar Common Definitions: *********
//
#define      TOKEN_LENGTH                2
#define      LENGTH_LENGTH               4


// #define   VARTYPE_x         Use Token Definitions for VarType

#define      IDTYPE_MSID                 11
#define      IDTYPE_MSGRPID              12
#define      IDTYPE_USERID               13
#define      IDTYPE_USERGRPID            14
#define      IDTYPE_LOGICAL              15
              // e.g. ip address and socket; e.g. inetd.cfg invocation (e.g. 23.56.232.2:34002)
#define      IDTYPE_PHYSICAL             16 // MS serial #

#define      ADDRTYPE_LOGICAL            21 // e.g. ip address
#define      ADDRTYPE_PHYSICAL           22 // e.g. MS serial #
#define      ADDRTYPE_POSTAL             23
#define      ADDRTYPE_POINT              24
#define      ADDRTYPE_SL                 25
#define      ADDRTYPE_2D                 26
#define      ADDRTYPE_3D                 27

#define      TIMESPECTYPE_STAMP          31
#define      TIMESPECTYPE_PERIOD         32
```

Fig. 34A

```
//********* Grammar Common Construct Token Definitions: *********
//
// #define    VARIABLES                  10001
#define       VARIABLE                   10002
// #define    VARINSTANTIATIONS          10003
#define       VARINSTANTIATE             10004
#define       VARNAME                    10005
#define       DESCRIPTION                10006
#define       HISTORY                    10007
#define       CREATORINFO                10008
#define       MODIFIERINFO               10009
#define       CREATEDATETIME             10010
#define       CREATORID                  10011
#define       CREATORIDTYPE              10012
#define       CREATORADDR                10013
#define       CREATORSYSID               10014
#define       CREATORSYSTYPE             10015
#define       CREATORSYSADDR             10016
#define       LASTMODIFYDATETIME         10017
#define       LASTMODIFYID               10018
#define       LASTMODIFYIDTYPE           10019
#define       LASTMODIFYADDR             10020
#define       LASTMODIFYSYSID            10021
#define       LASTMODIFYSYSTYPE          10022
#define       LASTMODIFYSYSADDR          10023
#define       ID                         10024
#define       IDTYPE                     10025
#define       ADDRESS                    10026
#define       TIMESPEC                   10027


//********* Grammar Permission Construct Token Definitions: *********
//
#define       PERMISSIONBODY             12001
// #define    PERMISSIONS                12002
#define       PERMISSION                 12003
#define       GRANTOR                    12004
#define       GRANTEE                    12005
#define       GRANTS                     12006
#define       GRANT                      12007
// #define    PRIVILEGES                 12008
#define       PRIVILEGE                  12009
#define       MSRELEVANCE                12010
```

Fig. 34B

```
#define      GROUPS                              12011
#define      GROUP                               12012
#define      IDS                                 12013

//********* Grammar Charter Construct Token Definitions: *********
//
#define      CHARTERBODY                         14001
// #define   CHARTERS                            14002
#define      CHARTER                             14003
#define      EXPRESSION                          14004
#define      CONDITIONS                          14005
#define      CONDOP                              14006
#define      CONDITION                           14007
#define      TERM                                14008
#define      WDRTERM                             14009
#define      APPTERM                             14010
#define      VALUE                               14011
#define      DATA                                14012
#define      INVOCATION                          14013
#define      OP                                  14014
// #define   ACTIONS                             14015
#define      ACTION                              14016
#define      HOST                                14017
#define      COMMAND                             14018
#define      OPERAND                             14019
// #define   PARAMETERS                          14020
#define      PARAMETER                           14021
#define      POINTSET                            14022

//********* Grammar Charter Definitions: *********
//
// atomic terms (e.g. \loc_my), WDR field terms (e.g. __location),
// Application terms (e.g. M_source), Invocation (e.g. fcn(p1,p2)), CondOp (e.g. "&") and
// Data atomic elements (e.g. c:\dir1\fname:58/LONGINT) are recognized syntaxes.
#define      VALUE_NUMBER                        41
#define      VALUE_TEXT                          42
#define      VALUE_ENUM                          43 // "value"
#define      VALUE_BOOLEAN                       44 // 1 = True, 0 = False
#define      VALUE_ID                            45

#define      DIMENSION2                          71
#define      DIMENSION3                          72
#define      FORMAT_GEO                          73
#define      FORMAT_CARTESIAN                    74
#define      FORMAT_POLAR                        75
```

Fig. 34C

```
//********* Atomic Commands : *********
//
#define         CMD_SEND                              101
#define         CMD_NOTIFY                            103
#define         CMD_COMPOSE                           105
#define         CMD_FIND                              107
#define         CMD_INVOKE                            109
#define         CMD_COPY                              111
#define         CMD_DISCARD                           113
#define         CMD_MOVE                              115
#define         CMD_STORE                             117
#define         CMD_CONNECT                           119
#define         CMD_ADMINISTRATE                      121
#define         CMD_CHANGE                            123

//********* Atomic Operands : *********
//
#define         OPERAND_AUTODIALNUMBER                201
#define         OPERAND_WEBLINK                       203
#define         OPERAND_EMAIL                         205
#define         OPERAND_SMSMSG                        207
#define         OPERAND_BRDEMAIL                      209
#define         OPERAND_BRDSMSMSG                     211
#define         OPERAND_INDICATOR                     213
#define         OPERAND_APP                           215
#define         OPERAND_DOCUMENT                      217
#define         OPERAND_FILE                          219
#define         OPERAND_CONTENT                       221
#define         OPERAND_DBOBJ                         223
#define         OPERAND_DATA                          225
#define         OPERAND_SEMAPHORE                     227
#define         OPERAND_DIRECTORY                     229
#define         OPERAND_APPCONTEXT                    231
#define         OPERAND_UIFOBJ                        233
#define         OPERAND_UIFCTL                        235
#define         OPERAND_INPUT                         237
#define         OPERAND_OUTPUT                        239
#define         OPERAND_ALERT                         241
#define         OPERAND_PROC                          243
#define         OPERAND_CONTAINER                     245
#define         OPERAND_PROGOBJ                       247
#define         OPERAND_CURSOR                        249
#define         OPERAND_CALENDAR                      251
#define         OPERAND_ADDRESSBOOK                   253
```

Fig. 34D

```
// TIMESPEC date/time stamps for open ended periods are set with no start/end spec:
//  >=YYYMMDDHHMMSS.1..J ==> set x.endDT to DT_NOENDSPEC;
//  <=YYYMMDDHHMMSS.1..J ==> set x.startDT to DT_NOSTARTSPEC;
// <YYYMMDDHHMMSS.1..J and >YYYMMDDHHMMSS.1..J subtracts/adds min precision
// from specified date/time stamp (i.e. TIMESPEC periods are preferably inclusive).
#define DT_NOENDSPEC            -1.0
#define DT_NOSTARTSPEC          -2.0

typedef struct timespec { // specifications converted to a Julian period form
            double              startDT;    // converted to Julian format (1ms precision)
            double              endDT;      //  converted to Julian format (1ms precision)
            struct timespec     *nextTS;    // linked list of sibling timespecs
            } TIMESPEC;

typedef struct {
            double              *dt;        // Julian date/time
            unsigned char       id[MAX_IDLENGTH];
            unsigned short      idtype;     // for IDTYPE_x values
            // unsigned short   cadr_type;  // Assume 1 format here
            unsigned char       *c_address;
            char                *sysid;
            char                *systype;
            // unsigned short   sysadr_type; // Assume 1 format here
            unsigned char       *sys_address;
            } BOOKKEEP;

typedef struct {
            BOOKKEEP *creation;
            BOOKKEEP *modify;
            } HISTRY;

typedef struct {
            unsigned short      vartype;
            char                name[MAX_VARNAME];
            unsigned char       *value;     // may be complex or series of complex
            } VAR;
```

Fig. 34E

```
typedef struct privilege {
          unsigned long      priv;          // constant value of known privilege
          unsigned char      relevance[MAX_RELEVANCEMASK];
          TIMESPEC           *tspec;
          char               *desc;
          HISTRY             *hist;
          struct privilege   *nextPriv;     // linked list of sibling privileges
          } PRIVILEGE;

typedef struct grant {
          char               name[MAX_GRNAMLENGTH];
          char               permtype;      // 'P' = Privilege(s), 'G' = Grant(s)
          union  {
               struct grant  *grants;       // linked list subordinate/descending grant(s)
               PRIVILEGE     *privileges;   // linked list of privilege(s)
               } assigned;
          TIMESPEC           *tspec;
          char               *desc;
          HISTRY             *hist;
          struct grant       *nextGrant;    // linked list of sibling grants
          } GRANT;

typedef struct permission {
          unsigned char      grantor[MAX_IDLENGTH];
          unsigned short     gortype;            // for IDTYPE_x values
          unsigned char      grantee[MAX_IDLENGTH];
          unsigned short     geetype;            // for IDTYPE_x values
          char               permtype;           // 'P' = Privilege(s), 'G' = Grant(s)
          union  {
               GRANT         *grants;            // linked list of grant(s)
               PRIVILEGE     *privileges;        // linked list of privilege(s)
               } assigned;
          TIMESPEC           *tspec;
          char               *desc;
          HISTRY             *hist;
          struct permission  *nextPerm;          // linked list of permissions
          } PERMISSION;
```

Fig. 34F

```
typedef struct identity {
            unsigned char       id[MAX_IDLENGTH];
            unsigned short      idtype;      // for IDTYPE_x values
            struct identity     *nextID;     // linked list of sibling IDs
            } IDENTITY;

typedef struct group {
            char                name[MAX_GRPNAMELENGTH];
            char                grptype;     // 'B' = Branch, 'L' = Leaf
            union  {
                  struct group  *groups;     // linked list subordinate/descending group(s)
                  IDENTITY      *ids;        // linked list of IDs in this group
                  } assigned;
            char                *desc;
            HISTRY              *hist;
            struct grant        *nextGroup;  // linked list of sibling groups
            } GROUP;

typedef struct action {
            IDENTITY            host;   // .idtype = 0 = no host spec)
            unsigned short      cmd;
            unsigned short      operand;
            unsigned char       *params;           // maintained in syntax for flexibility
                                                   // and for stack processing
            TIMESPEC            *tspec;
            char                *desc;
            HISTRY              *hist;
            struct action       *nextActn;         // linked list of sibling actions
            } ACTION;

typedef struct charter {
            unsigned char       grantee[MAX_IDLENGTH];
            unsigned short      geetype;           // for IDTYPE_x values
            unsigned char       grantor[MAX_IDLENGTH];
            unsigned short      gortype;           // for IDTYPE_x values
            TIMESPEC            *exprTS;
            unsigned char       *cond;             // at least 1 condition maintained in
                                                   // syntax for proper stack processing
            ACTION              *actn;
            char                *desc;
            HISTRY              *hist;
            struct charter      *nextCharter;      // linked list of charters
            } CHARTER;
```

Fig. 34G

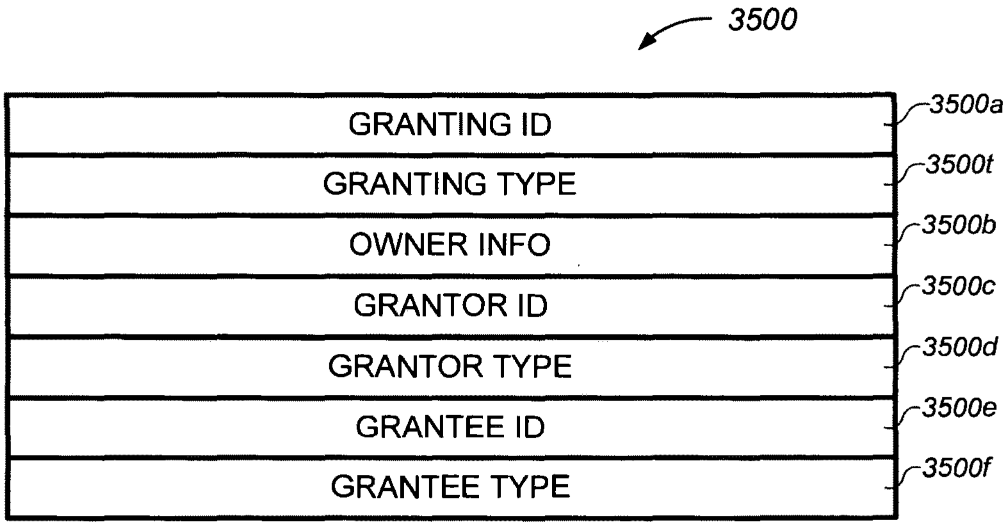

3500
GRANTING ID
3500a
GRANTING TYPE
3500t
OWNER INFO
3500b
GRANTOR ID
3500c
GRANTOR TYPE
3500d
GRANTEE ID
3500e
GRANTEE TYPE
3500f

Fig. 35A

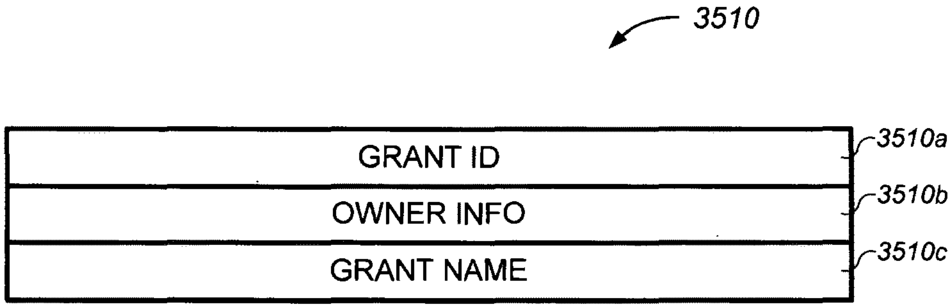


Fig. 35B

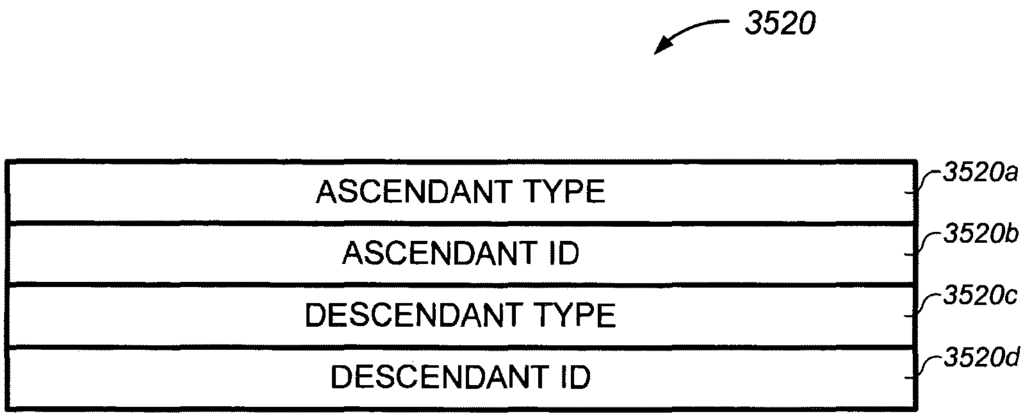

3520
ASCENDANT TYPE
3520a
ASCENDANT ID
3520b
DESCENDANT TYPE
3520c
DESCENDANT ID
3520d

Fig. 35C

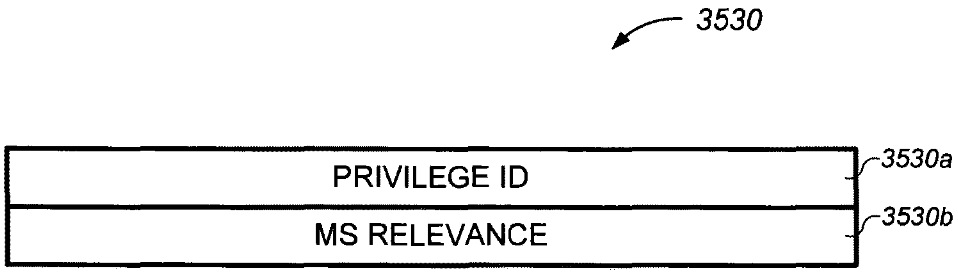


Fig. 35D

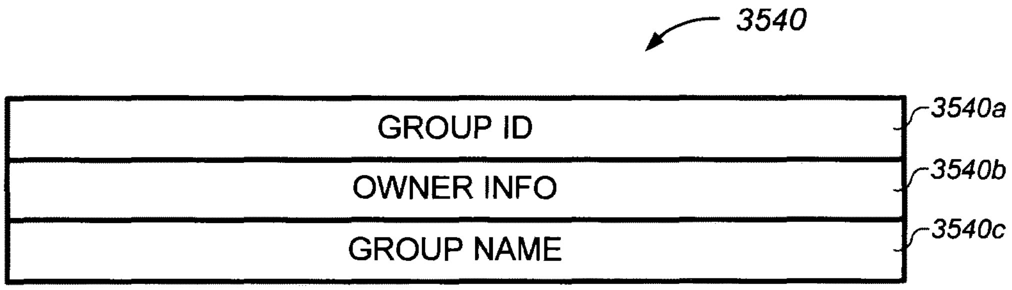


Fig. 35E

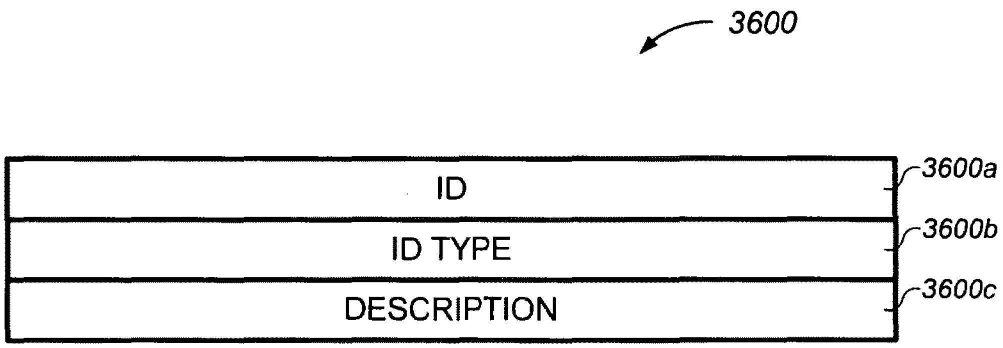


Fig. 36A

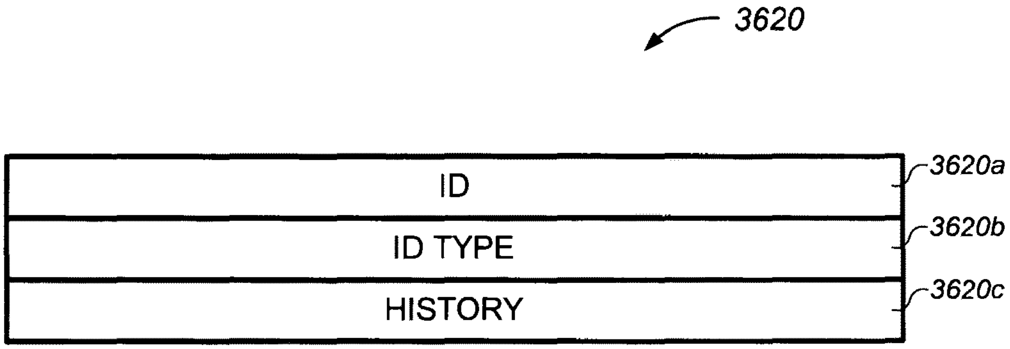


Fig. 36B

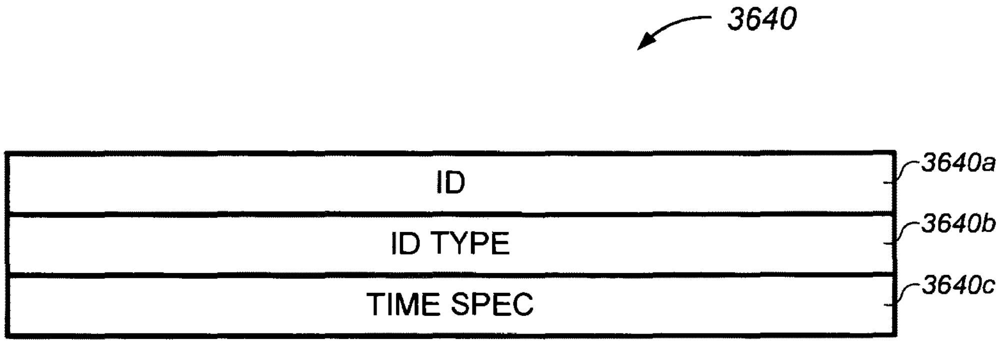


Fig. 36C

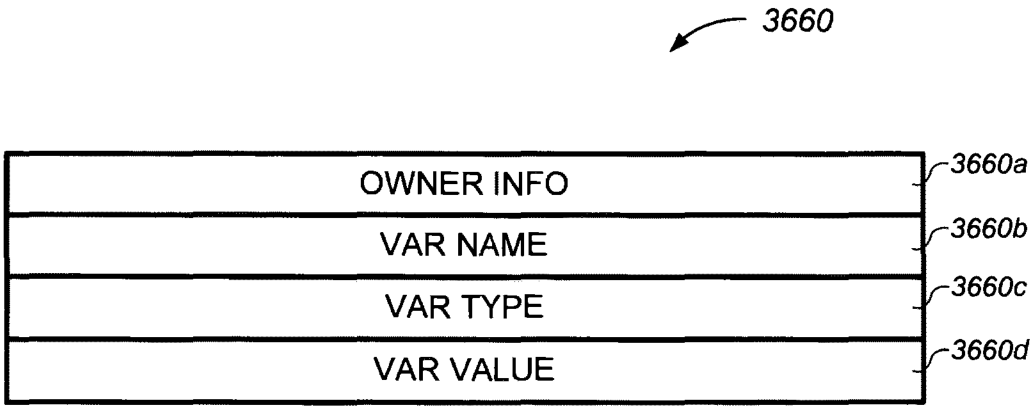

3660
OWNER INFO
VAR NAME
VAR TYPE
VAR VALUE
3660a
3660b
3660c
3660d

Fig. 36D

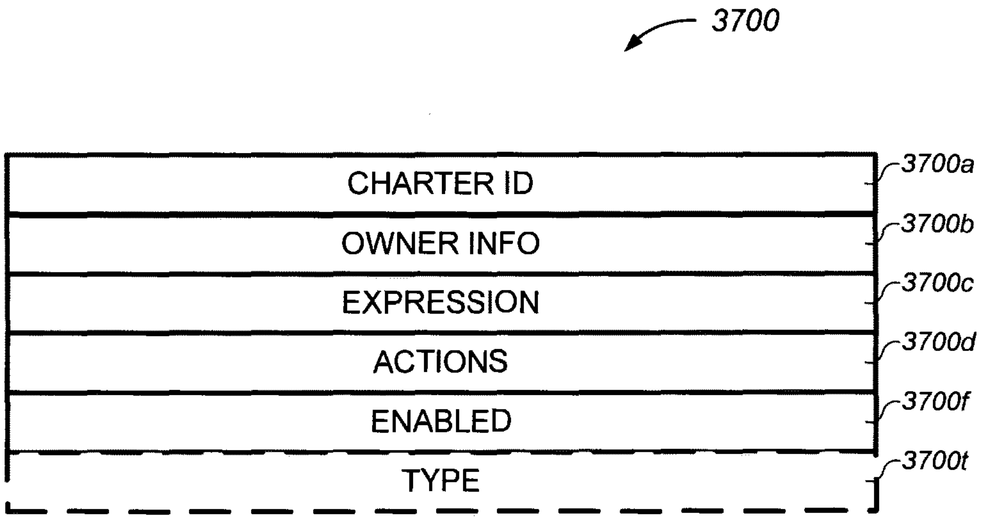

3700
CHARTER ID
3700a
OWNER INFO
3700b
EXPRESSION
3700c
ACTIONS
3700d
ENABLED
3700f
TYPE
3700t

Fig. 37A

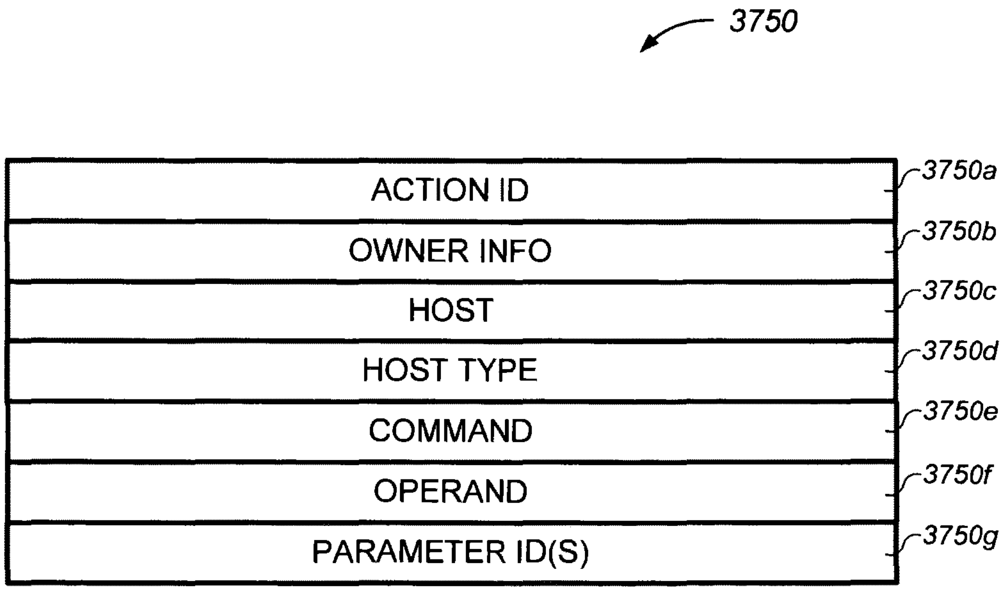

3750
ACTION ID
3750a
OWNER INFO
3750b
HOST
3750c
HOST TYPE
3750d
COMMAND
3750e
OPERAND
3750f
PARAMETER ID(S)
3750g

Fig. 37B

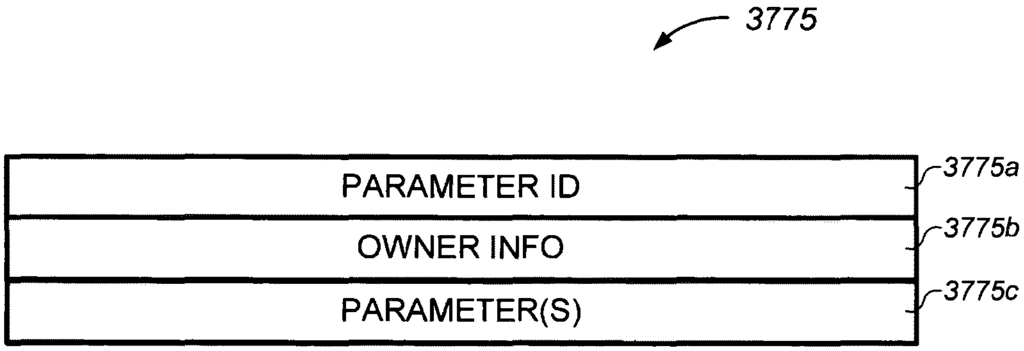


Fig. 37C

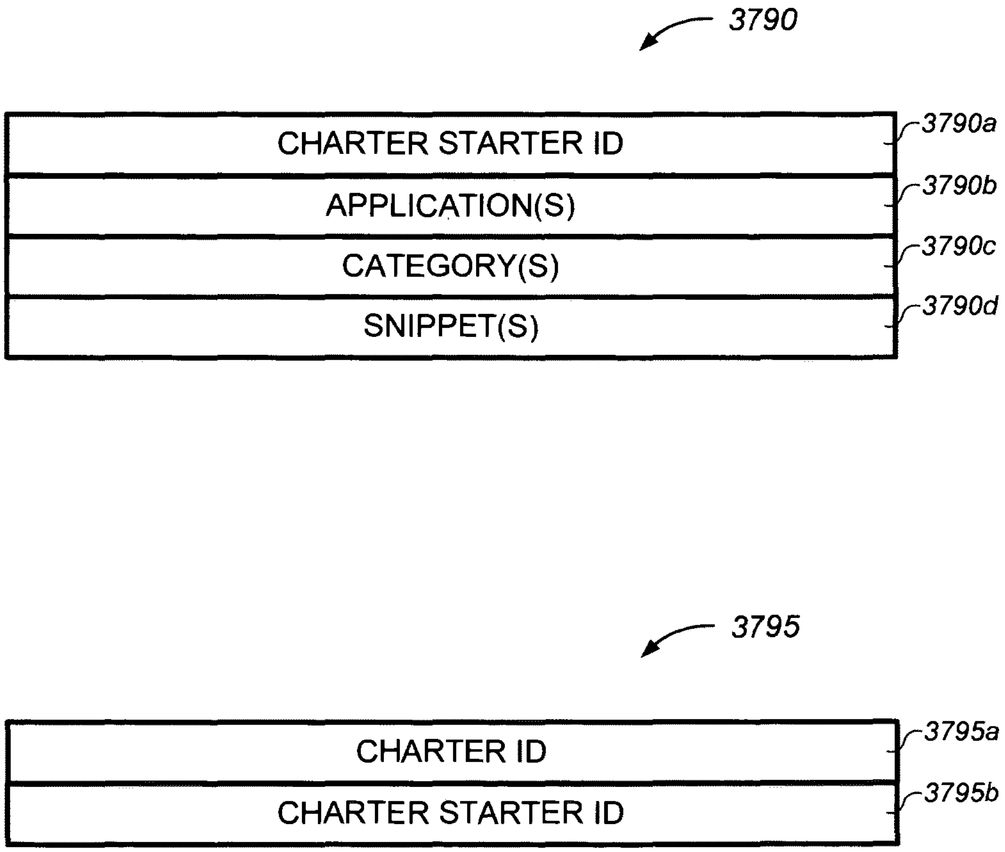

3790
CHARTER STARTER ID
3790a
APPLICATION(S)
3790b
CATEGORY(S)
3790c
SNIPPET(S)
3790d
3795
CHARTER ID
3795a
CHARTER STARTER ID
3795b

Fig. 37D

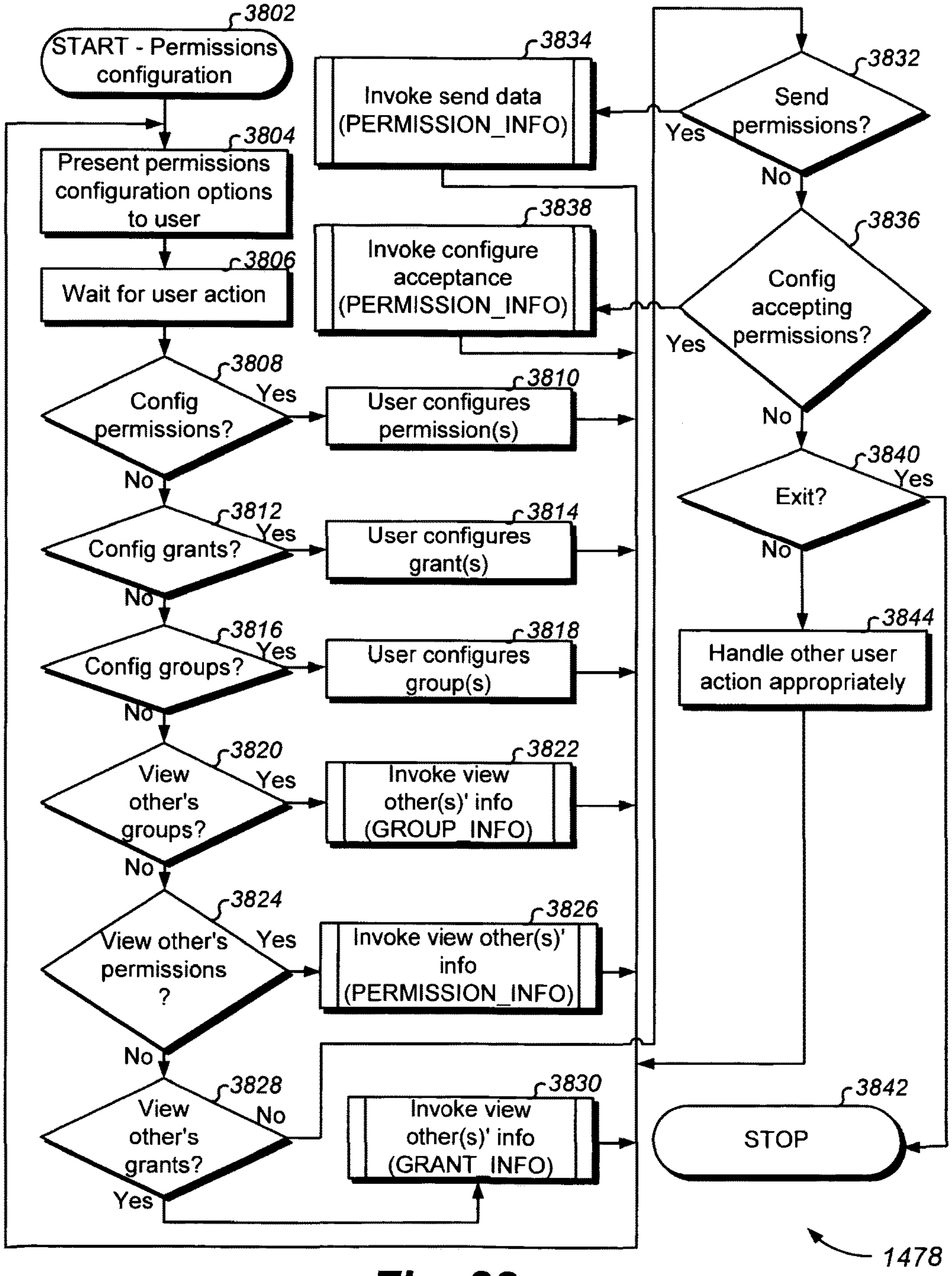

3802
START - Permissions configuration
3804
Present permissions configuration options to user
3806
Wait for user action
3808
Config permissions?
Yes
No
3810
User configures permission(s)
3812
Config grants?
Yes
No
3814
User configures grant(s)
3816
Config groups?
Yes
No
3818
User configures group(s)
3820
View other's groups?
Yes
No
3822
Invoke view other(s)' info (GROUP_INFO)
3824
View other's permissions ?
Yes
No
3826
Invoke view other(s)' info (PERMISSION_INFO)
3828
View other's grants?
No
Yes
3830
Invoke view other(s)' info (GRANT_INFO)
3834
Invoke send data (PERMISSION_INFO)
3838
Invoke configure acceptance (PERMISSION_INFO)
3832
Send permissions?
Yes
No
3836
Config accepting permissions?
Yes
No
3840
Exit?
Yes
No
3844
Handle other user action appropriately
3842
STOP
1478

Fig. 38

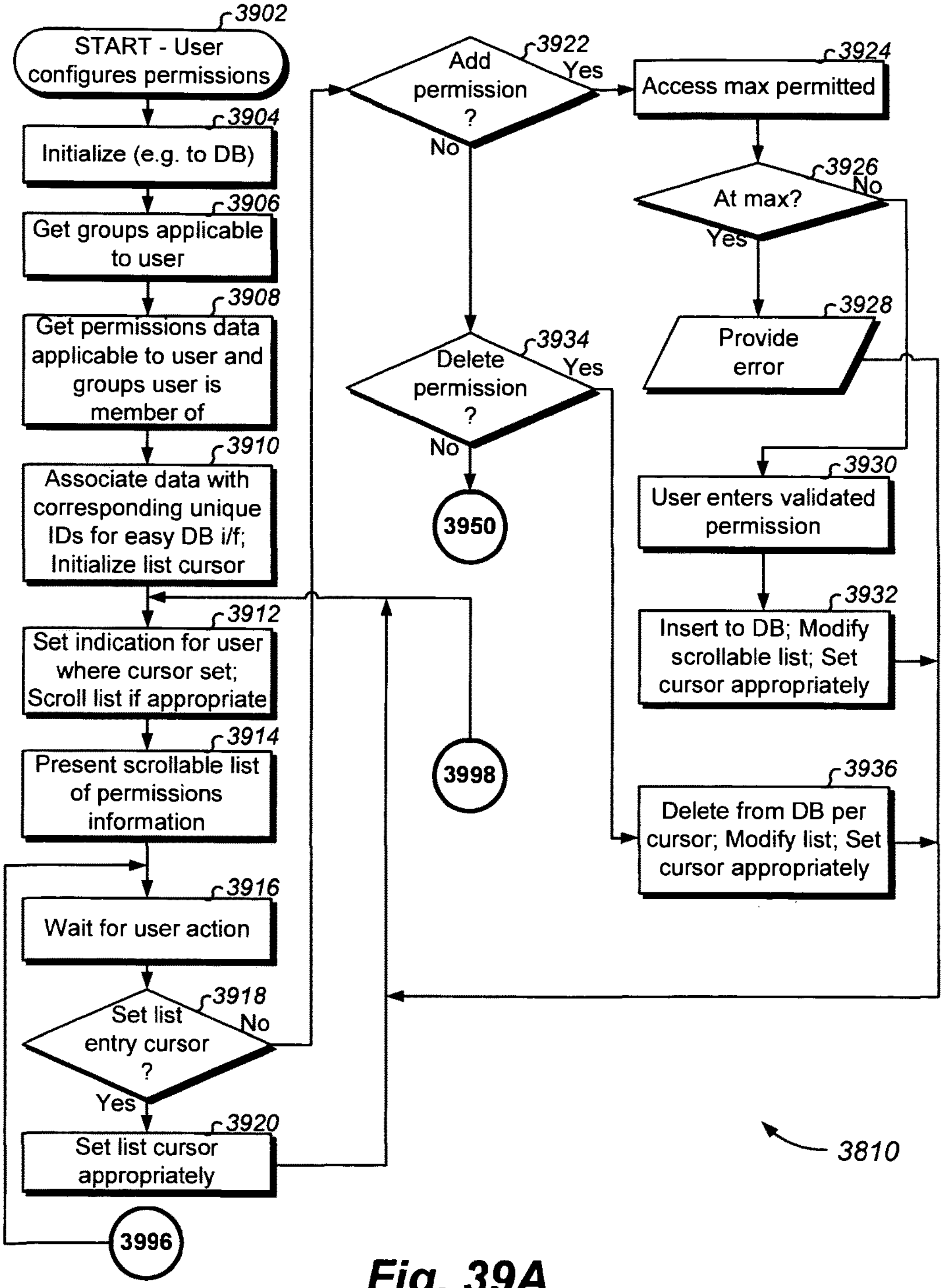

3902
START - User configures permissions
3904
Initialize (e.g. to DB)
3906
Get groups applicable to user
3908
Get permissions data applicable to user and groups user is member of
3910
Associate data with corresponding unique IDs for easy DB i/f; Initialize list cursor
3912
Set indication for user where cursor set; Scroll list if appropriate
3914
Present scrollable list of permissions information
3916
Wait for user action
3918
Set list entry cursor ?
No
Yes
3920
Set list cursor appropriately
3996
3922
Add permission ?
Yes
No
3924
Access max permitted
3926
At max?
No
Yes
3928
Provide error
3934
Delete permission ?
Yes
No
3950
3998
3930
User enters validated permission
3932
Insert to DB; Modify scrollable list; Set cursor appropriately
3936
Delete from DB per cursor; Modify list; Set cursor appropriately
3810

Fig. 39A

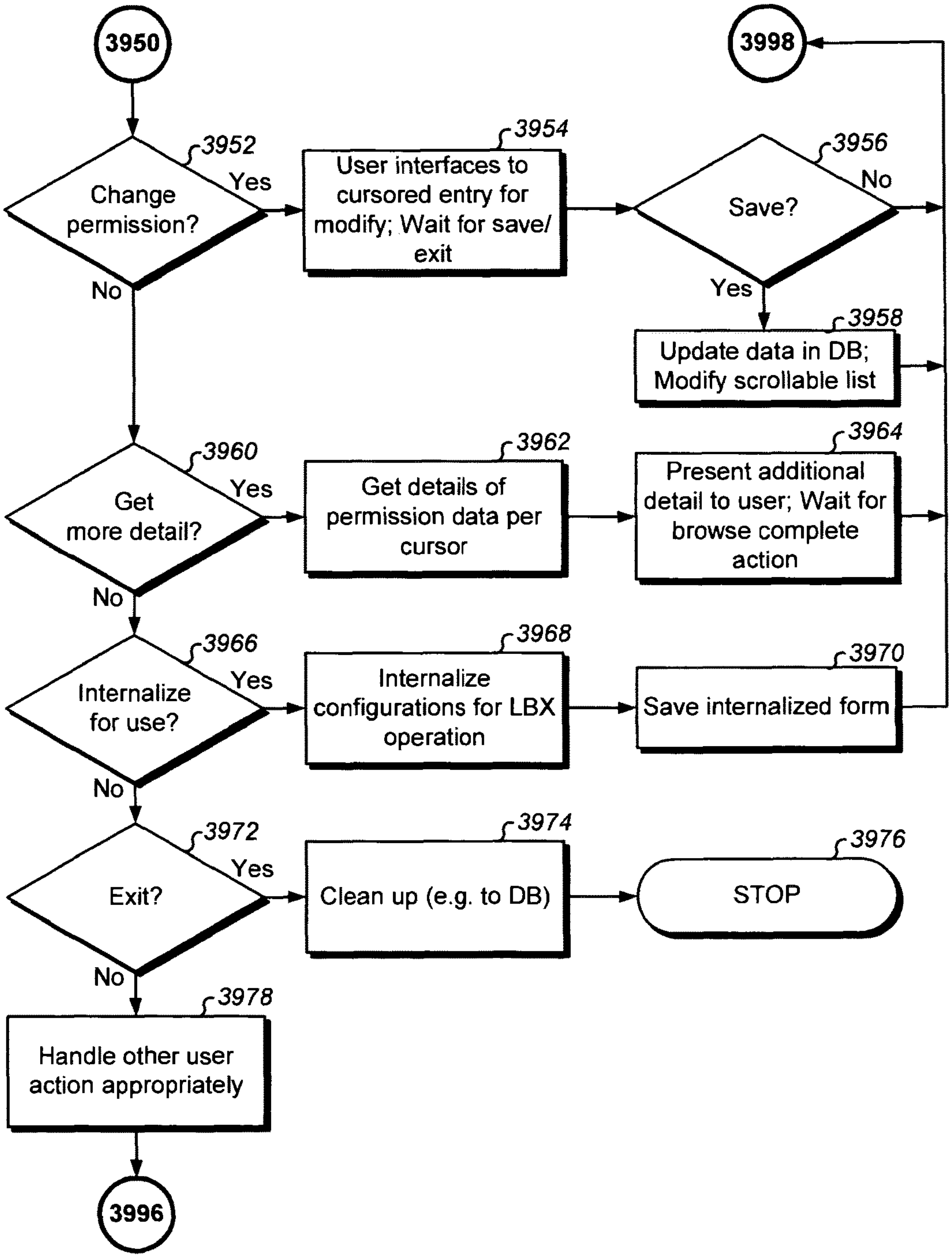

3950
3998
Change permission?
3952
Yes
No
User interfaces to cursored entry for modify; Wait for save/ exit
3954
Save?
3956
No
Yes
Update data in DB; Modify scrollable list
3958
Get more detail?
3960
Yes
No
Get details of permission data per cursor
3962
Present additional detail to user; Wait for browse complete action
3964
Internalize for use?
3966
Yes
No
Internalize configurations for LBX operation
3968
Save internalized form
3970
Exit?
3972
Yes
No
Clean up (e.g. to DB)
3974
STOP
3976
Handle other user action appropriately
3978
3996

Fig. 39B

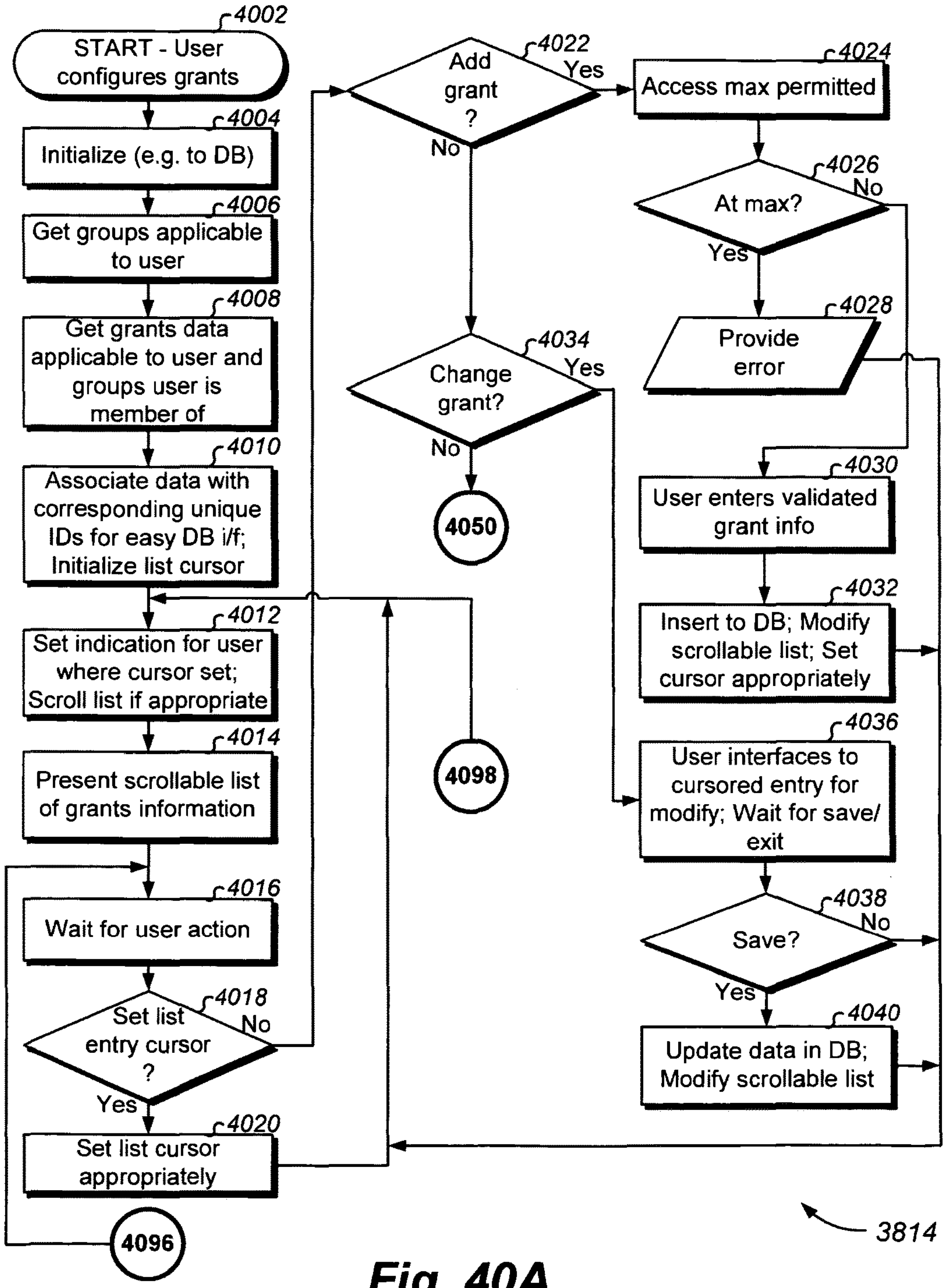

4002
START - User configures grants
4004
Initialize (e.g. to DB)
4006
Get groups applicable to user
4008
Get grants data applicable to user and groups user is member of
4010
Associate data with corresponding unique IDs for easy DB i/f; Initialize list cursor
4012
Set indication for user where cursor set; Scroll list if appropriate
4014
Present scrollable list of grants information
4016
Wait for user action
4018
Set list entry cursor ?
No
Yes
4020
Set list cursor appropriately
4096
4022
Add grant ?
Yes
No
4024
Access max permitted
4026
At max?
No
Yes
4028
Provide error
4034
Change grant?
Yes
No
4050
4098
4030
User enters validated grant info
4032
Insert to DB; Modify scrollable list; Set cursor appropriately
4036
User interfaces to cursored entry for modify; Wait for save/ exit
4038
Save?
No
Yes
4040
Update data in DB; Modify scrollable list
3814

Fig. 40A

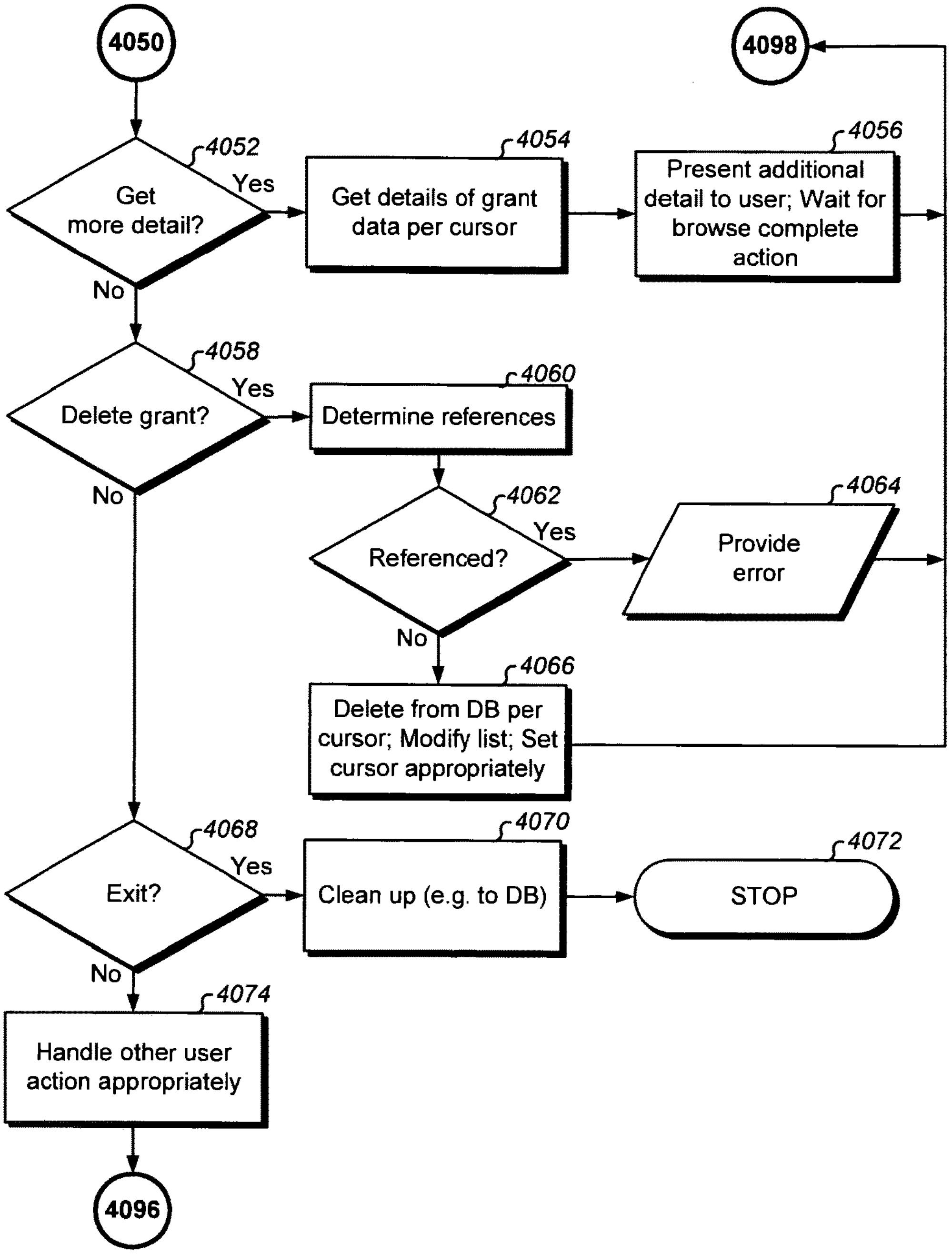

4050
4098
4052
Get more detail?
Yes
No
4054
Get details of grant data per cursor
4056
Present additional detail to user; Wait for browse complete action
4058
Delete grant?
Yes
No
4060
Determine references
4062
Referenced?
Yes
No
4064
Provide error
4066
Delete from DB per cursor; Modify list; Set cursor appropriately
4068
Exit?
Yes
No
4070
Clean up (e.g. to DB)
4072
STOP
4074
Handle other user action appropriately
4096

Fig. 40B

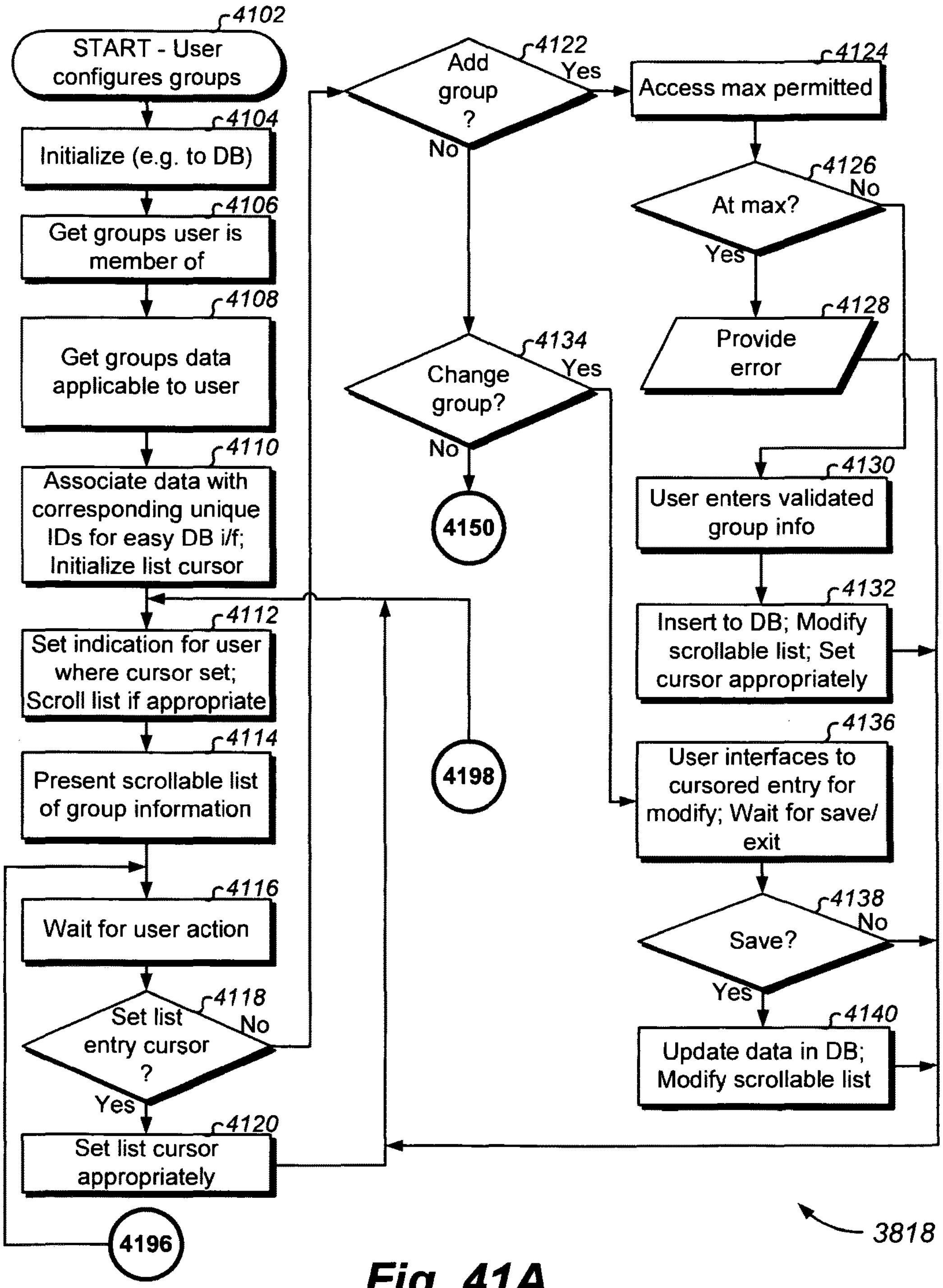

4102
START - User configures groups
4104
Initialize (e.g. to DB)
4106
Get groups user is member of
4108
Get groups data applicable to user
4110
Associate data with corresponding unique IDs for easy DB i/f; Initialize list cursor
4112
Set indication for user where cursor set; Scroll list if appropriate
4114
Present scrollable list of group information
4116
Wait for user action
4118
Set list entry cursor ?
No
Yes
4120
Set list cursor appropriately
4196
4122
Add group ?
Yes
No
4124
Access max permitted
4126
At max?
No
Yes
4128
Provide error
4134
Change group?
Yes
No
4150
4198
4130
User enters validated group info
4132
Insert to DB; Modify scrollable list; Set cursor appropriately
4136
User interfaces to cursored entry for modify; Wait for save/ exit
4138
Save?
No
Yes
4140
Update data in DB; Modify scrollable list
3818

Fig. 41A

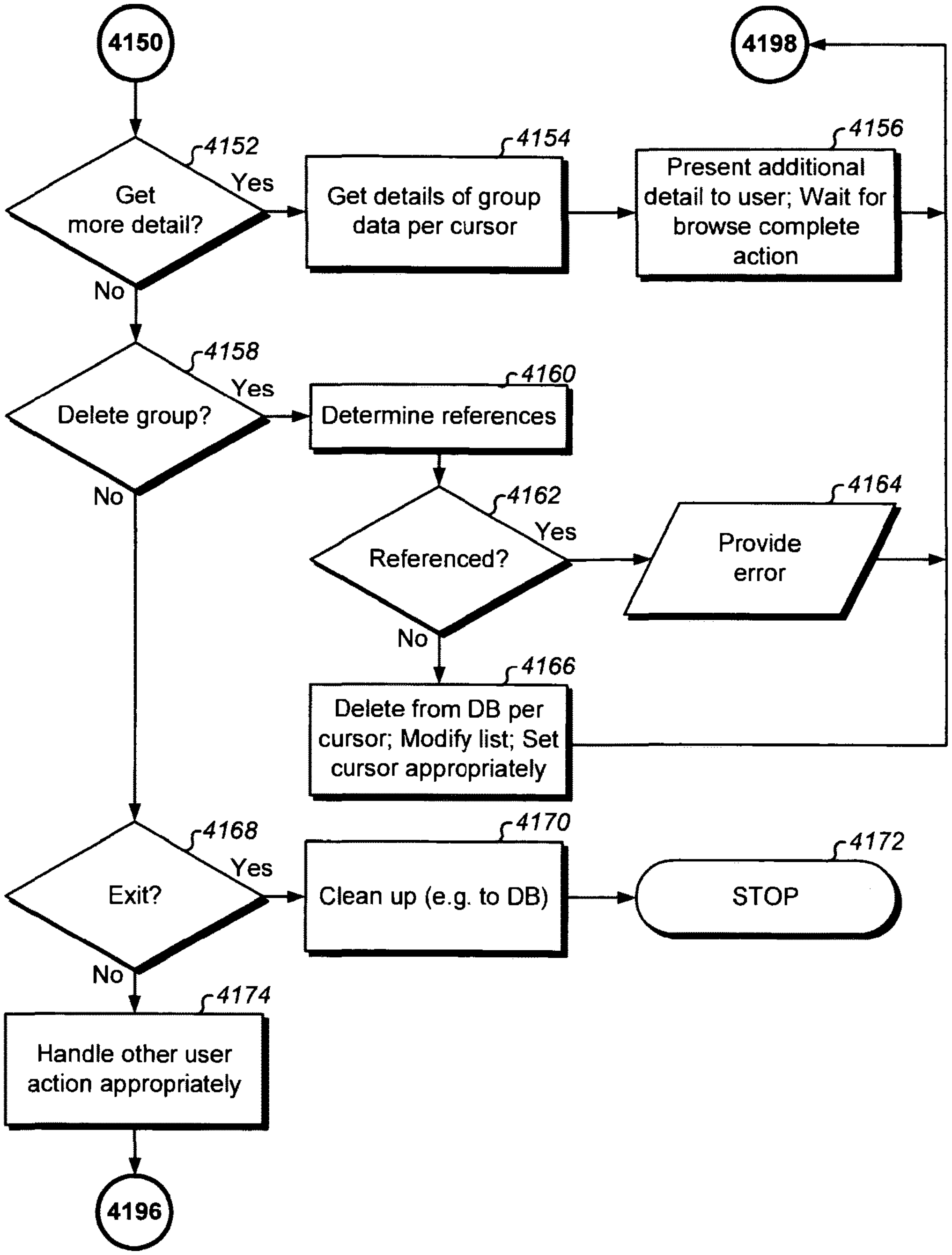

4150
4198
4152
Get more detail?
Yes
No
4154
Get details of group data per cursor
4156
Present additional detail to user; Wait for browse complete action
4158
Delete group?
Yes
No
4160
Determine references
4162
Referenced?
Yes
No
4164
Provide error
4166
Delete from DB per cursor; Modify list; Set cursor appropriately
4168
Exit?
Yes
No
4170
Clean up (e.g. to DB)
4172
STOP
4174
Handle other user action appropriately
4196

Fig. 41B

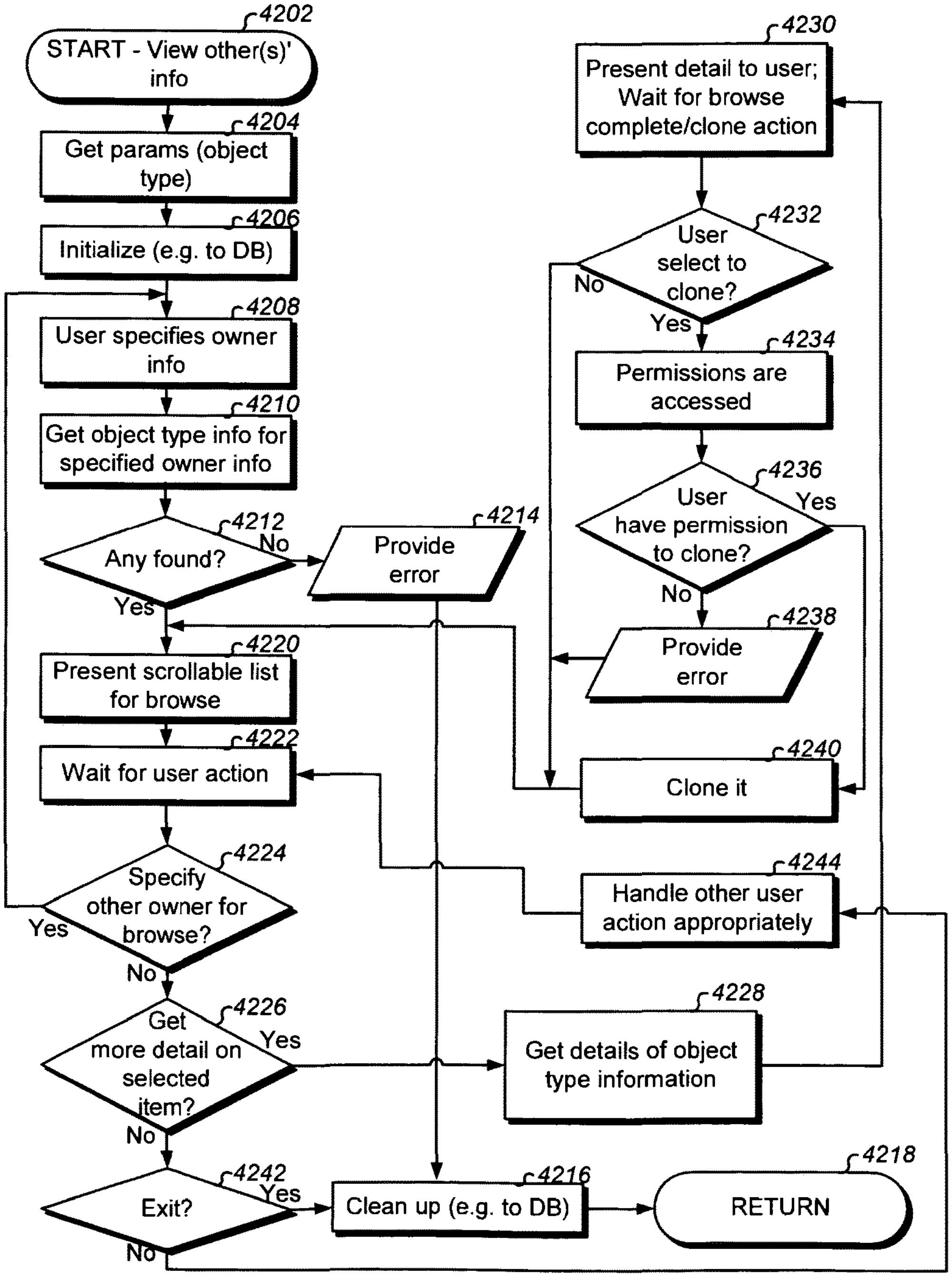

4202
START - View other(s)' info
4204
Get params (object type)
4206
Initialize (e.g. to DB)
4208
User specifies owner info
4210
Get object type info for specified owner info
4212
Any found?
No
Yes
4214
Provide error
4220
Present scrollable list for browse
4222
Wait for user action
4224
Specify other owner for browse?
Yes
No
4226
Get more detail on selected item?
Yes
No
4242
Exit?
Yes
No
4216
Clean up (e.g. to DB)
4218
RETURN
4228
Get details of object type information
4230
Present detail to user; Wait for browse complete/clone action
4232
User select to clone?
No
Yes
4234
Permissions are accessed
4236
User have permission to clone?
Yes
No
4238
Provide error
4240
Clone it
4244
Handle other user action appropriately

Fig. 42

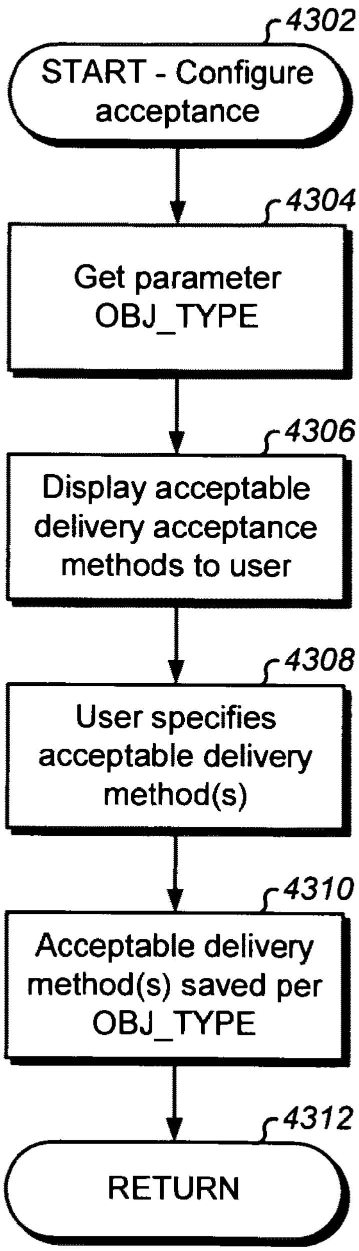

4302
START - Configure acceptance
4304
Get parameter OBJ_TYPE
4306
Display acceptable delivery acceptance methods to user
4308
User specifies acceptable delivery method(s)
4310
Acceptable delivery method(s) saved per OBJ_TYPE
4312
RETURN

Fig. 43

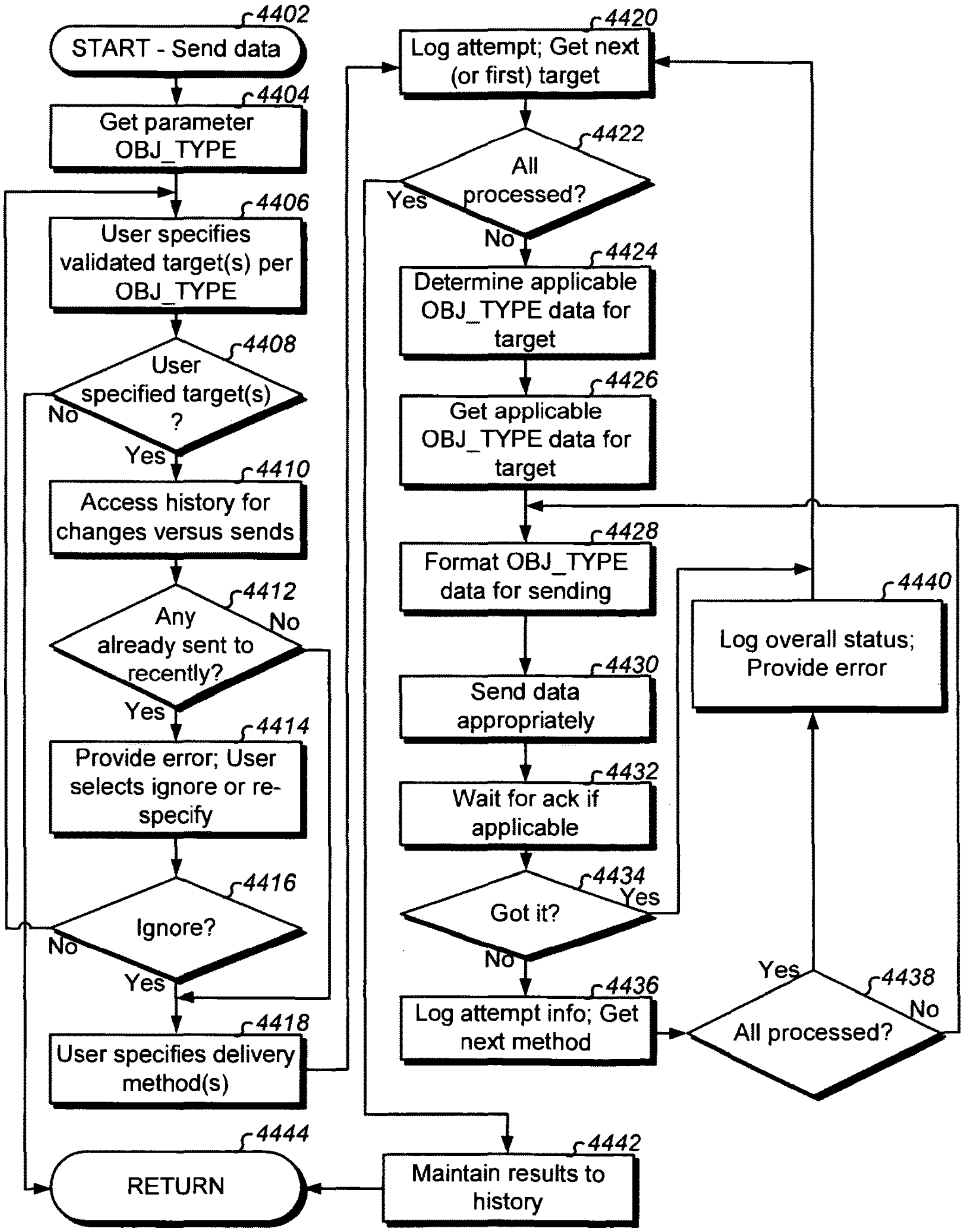

4402
START - Send data
4404
Get parameter OBJ_TYPE
4406
User specifies validated target(s) per OBJ_TYPE
4408
User specified target(s) ?
No
Yes
4410
Access history for changes versus sends
4412
Any already sent to recently?
No
Yes
4414
Provide error; User selects ignore or re-specify
4416
Ignore?
No
Yes
4418
User specifies delivery method(s)
4444
RETURN
4420
Log attempt; Get next (or first) target
4422
All processed?
Yes
No
4424
Determine applicable OBJ_TYPE data for target
4426
Get applicable OBJ_TYPE data for target
4428
Format OBJ_TYPE data for sending
4430
Send data appropriately
4432
Wait for ack if applicable
4434
Got it?
Yes
No
4436
Log attempt info; Get next method
4438
All processed?
Yes
No
4440
Log overall status; Provide error
4442
Maintain results to history

Fig. 44A

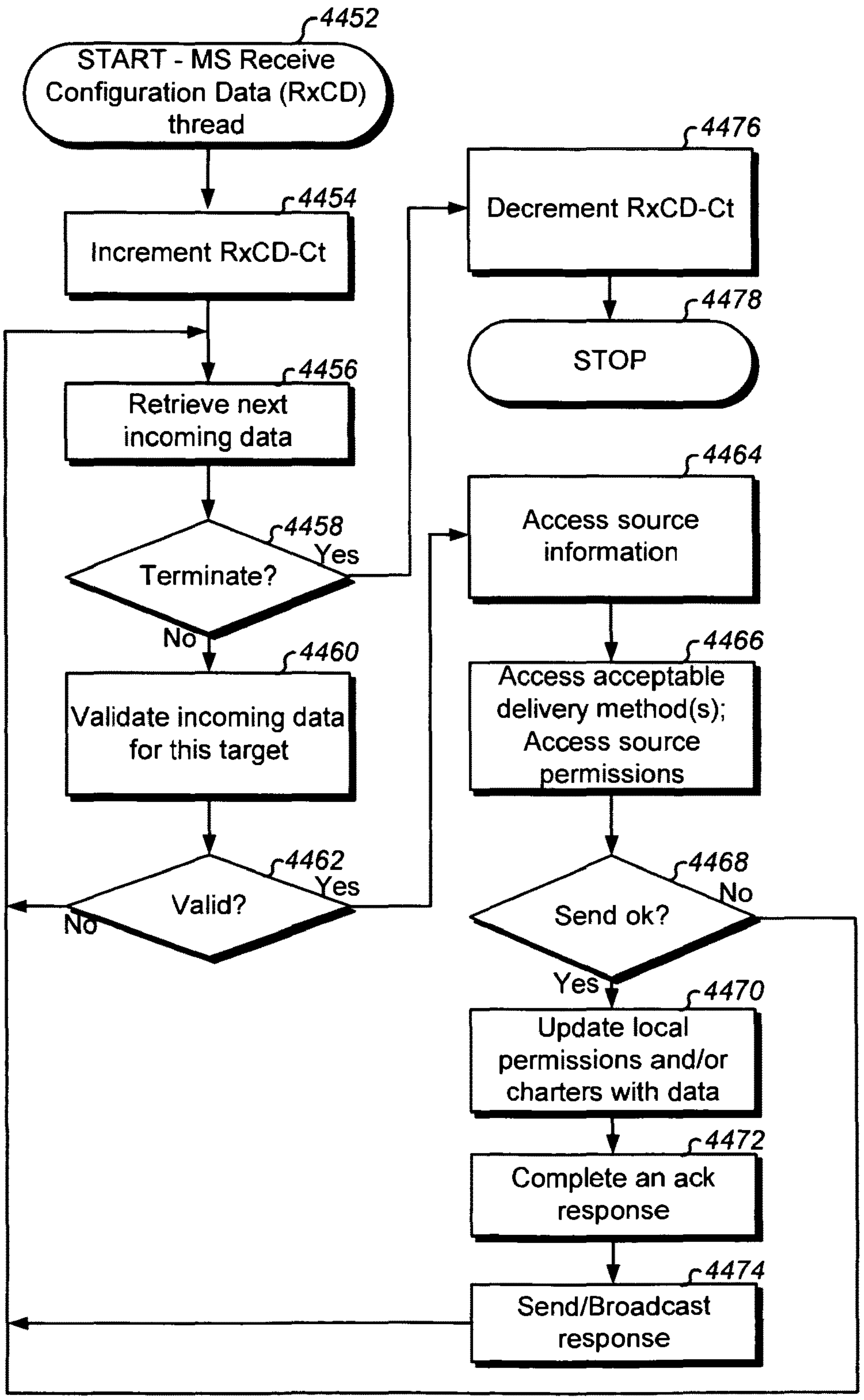

4452
START - MS Receive Configuration Data (RxCD) thread
4454
Increment RxCD-Ct
4456
Retrieve next incoming data
4458
Terminate?
Yes
No
4460
Validate incoming data for this target
4462
Valid?
Yes
No
4476
Decrement RxCD-Ct
4478
STOP
4464
Access source information
4466
Access acceptable delivery method(s); Access source permissions
4468
Send ok?
No
Yes
4470
Update local permissions and/or charters with data
4472
Complete an ack response
4474
Send/Broadcast response

Fig. 44B

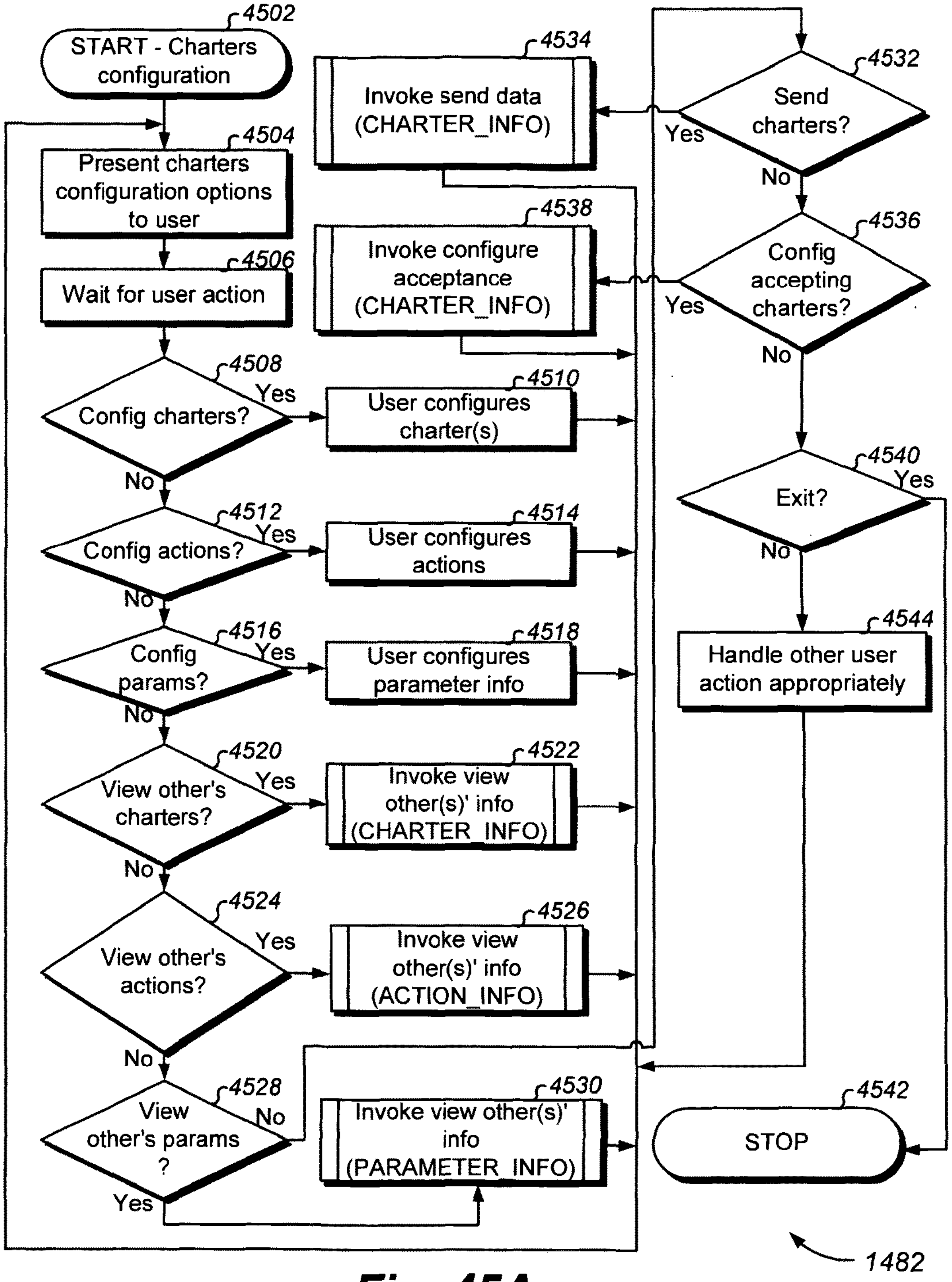

4502
START - Charters configuration
4504
Present charters configuration options to user
4506
Wait for user action
4508
Config charters?
Yes
No
4510
User configures charter(s)
4512
Config actions?
Yes
No
4514
User configures actions
4516
Config params?
Yes
No
4518
User configures parameter info
4520
View other's charters?
Yes
No
4522
Invoke view other(s)' info (CHARTER_INFO)
4524
View other's actions?
Yes
No
4526
Invoke view other(s)' info (ACTION_INFO)
4528
View other's params ?
No
Yes
4530
Invoke view other(s)' info (PARAMETER_INFO)
4534
Invoke send data (CHARTER_INFO)
4538
Invoke configure acceptance (CHARTER_INFO)
4532
Send charters?
Yes
No
4536
Config accepting charters?
Yes
No
4540
Exit?
Yes
No
4544
Handle other user action appropriately
4542
STOP
1482

Fig. 45A

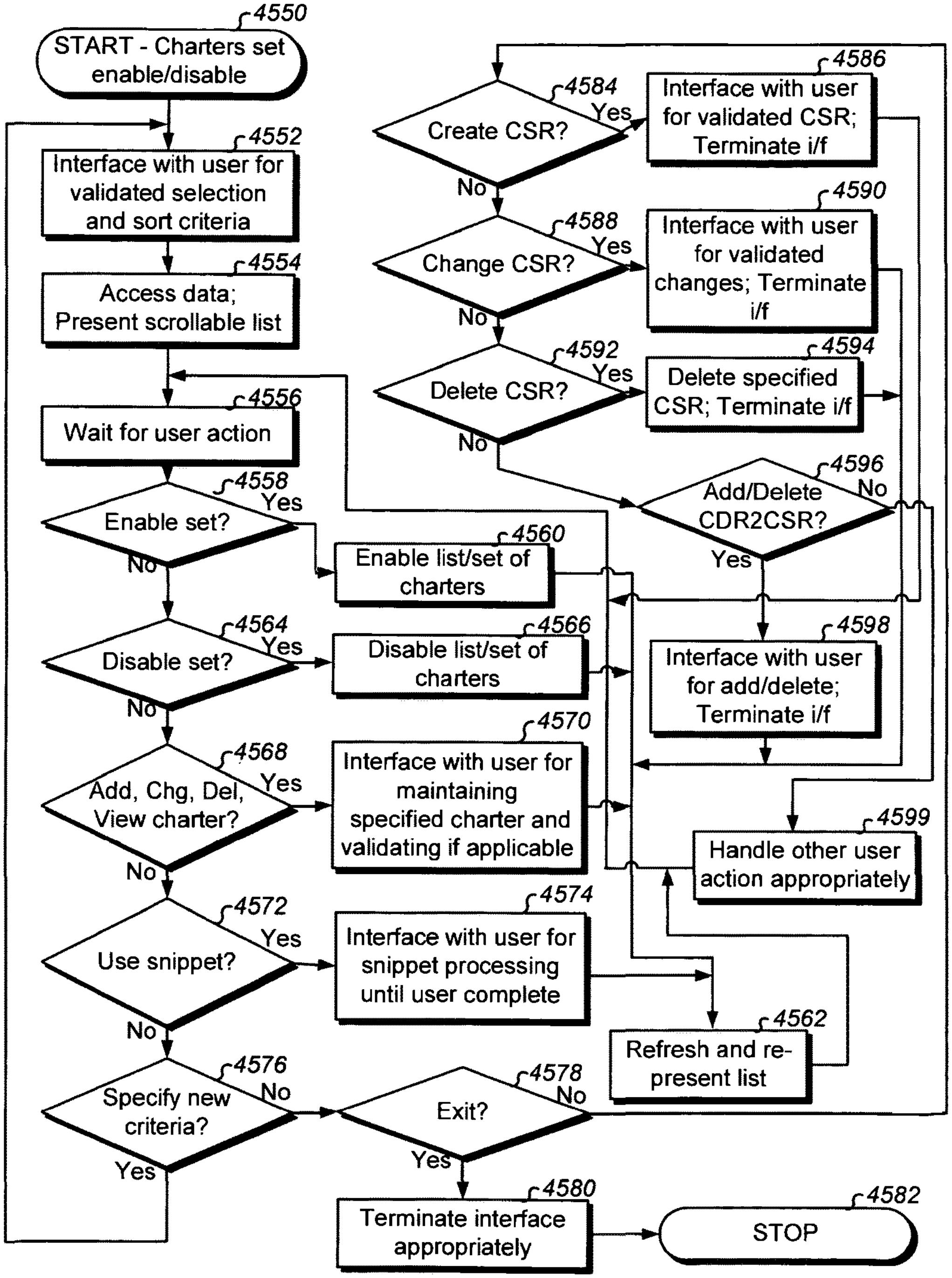

4550
START - Charters set enable/disable
4552
Interface with user for validated selection and sort criteria
4554
Access data; Present scrollable list
4556
Wait for user action
4558
Enable set?
Yes
No
4560
Enable list/set of charters
4564
Disable set?
Yes
No
4566
Disable list/set of charters
4568
Add, Chg, Del, View charter?
Yes
No
4570
Interface with user for maintaining specified charter and validating if applicable
4572
Use snippet?
Yes
No
4574
Interface with user for snippet processing until user complete
4576
Specify new criteria?
No
Yes
4578
Exit?
No
Yes
4580
Terminate interface appropriately
4582
STOP
4562
Refresh and re-present list
4584
Create CSR?
Yes
No
4586
Interface with user for validated CSR; Terminate i/f
4588
Change CSR?
Yes
No
4590
Interface with user for validated changes; Terminate i/f
4592
Delete CSR?
Yes
No
4594
Delete specified CSR; Terminate i/f
4596
Add/Delete CDR2CSR?
No
Yes
4598
Interface with user for add/delete; Terminate i/f
4599
Handle other user action appropriately

Fig. 45B

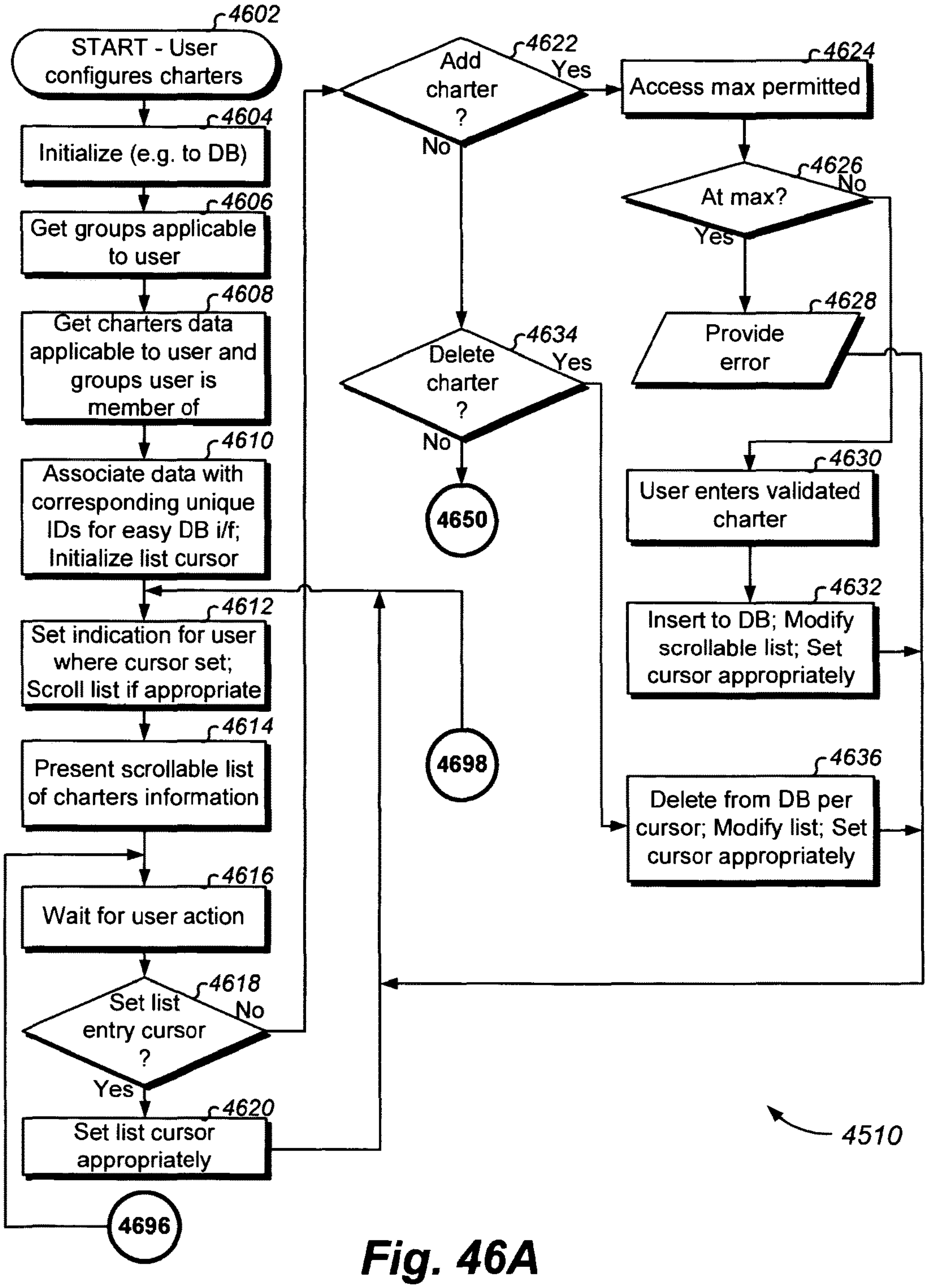

4602
START - User configures charters
4604
Initialize (e.g. to DB)
4606
Get groups applicable to user
4608
Get charters data applicable to user and groups user is member of
4610
Associate data with corresponding unique IDs for easy DB i/f; Initialize list cursor
4612
Set indication for user where cursor set; Scroll list if appropriate
4614
Present scrollable list of charters information
4616
Wait for user action
4618
Set list entry cursor ?
No
Yes
4620
Set list cursor appropriately
4696
4622
Add charter ?
Yes
No
4624
Access max permitted
4626
At max?
No
Yes
4628
Provide error
4634
Delete charter ?
Yes
No
4650
4698
4630
User enters validated charter
4632
Insert to DB; Modify scrollable list; Set cursor appropriately
4636
Delete from DB per cursor; Modify list; Set cursor appropriately
4510

Fig. 46A

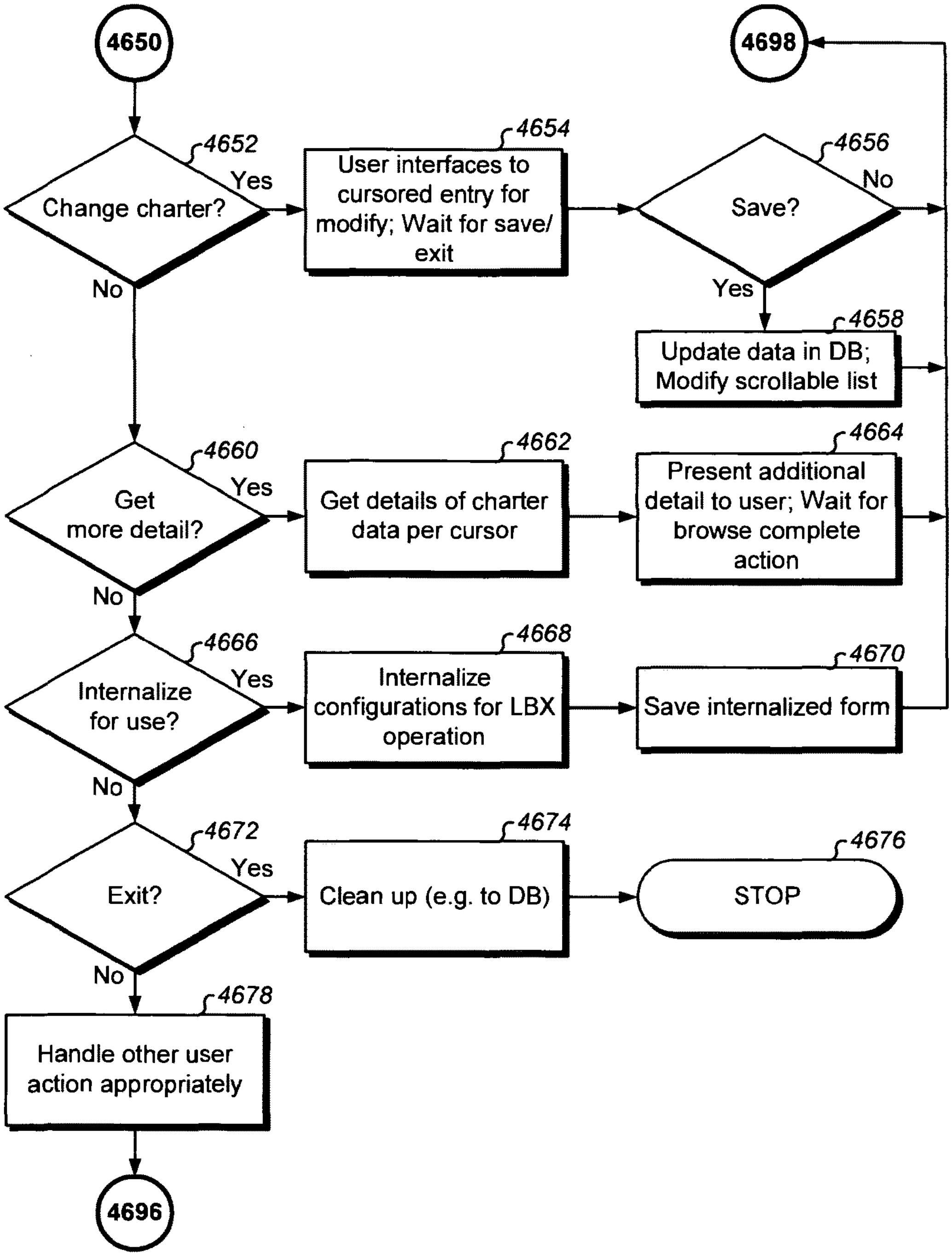

4650
4698
4652
Change charter?
Yes
No
4654
User interfaces to cursored entry for modify; Wait for save/ exit
4656
Save?
No
Yes
4658
Update data in DB; Modify scrollable list
4660
Get more detail?
Yes
No
4662
Get details of charter data per cursor
4664
Present additional detail to user; Wait for browse complete action
4666
Internalize for use?
Yes
No
4668
Internalize configurations for LBX operation
4670
Save internalized form
4672
Exit?
Yes
No
4674
Clean up (e.g. to DB)
4676
STOP
4678
Handle other user action appropriately
4696

Fig. 46B

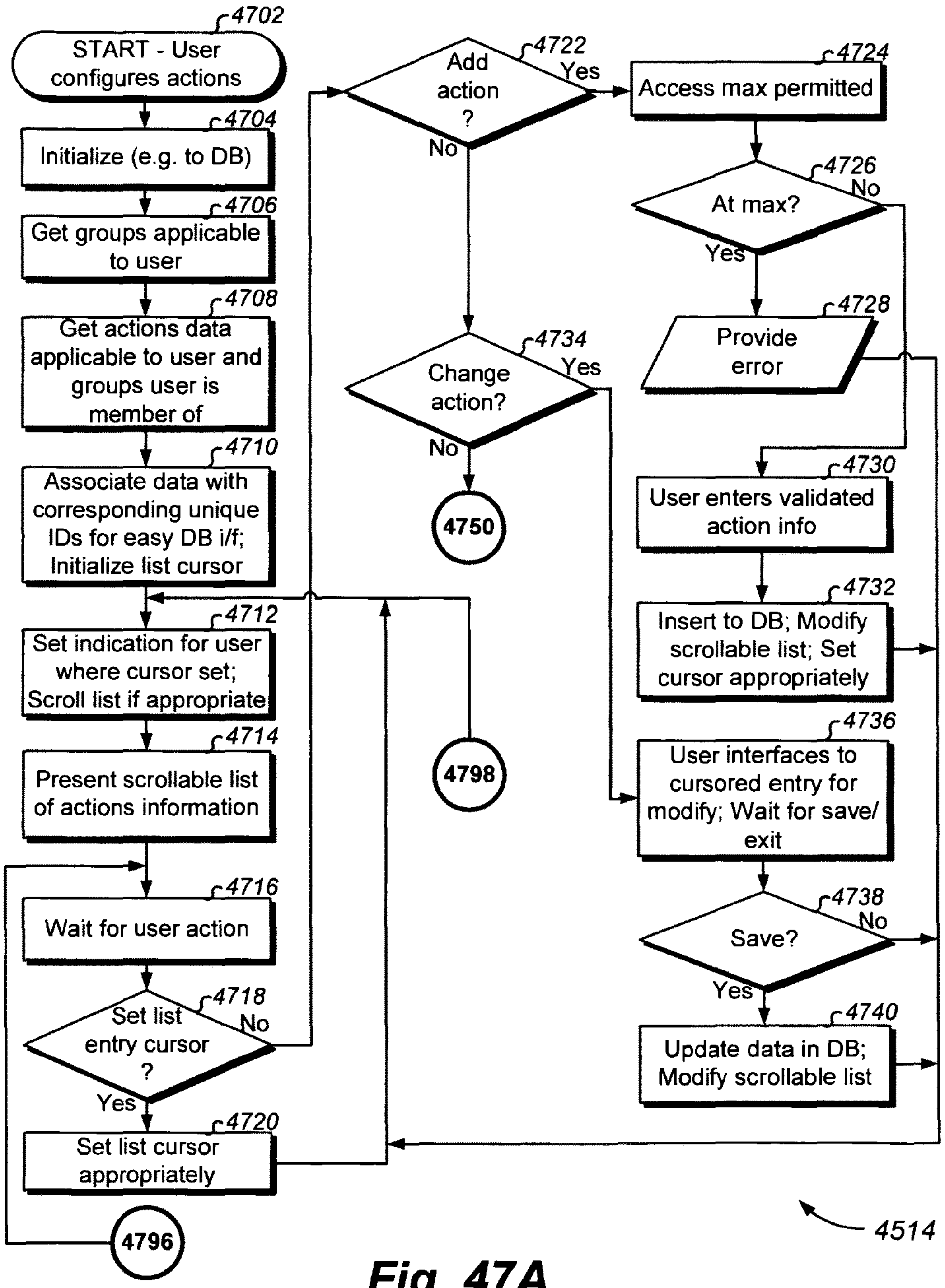

4702
START - User configures actions
4704
Initialize (e.g. to DB)
4706
Get groups applicable to user
4708
Get actions data applicable to user and groups user is member of
4710
Associate data with corresponding unique IDs for easy DB i/f; Initialize list cursor
4712
Set indication for user where cursor set; Scroll list if appropriate
4714
Present scrollable list of actions information
4716
Wait for user action
4718
Set list entry cursor ?
No
Yes
4720
Set list cursor appropriately
4796
4722
Add action ?
Yes
No
4724
Access max permitted
4726
At max?
No
Yes
4728
Provide error
4734
Change action?
Yes
No
4750
4798
4730
User enters validated action info
4732
Insert to DB; Modify scrollable list; Set cursor appropriately
4736
User interfaces to cursored entry for modify; Wait for save/ exit
4738
Save?
No
Yes
4740
Update data in DB; Modify scrollable list
4514

Fig. 47A

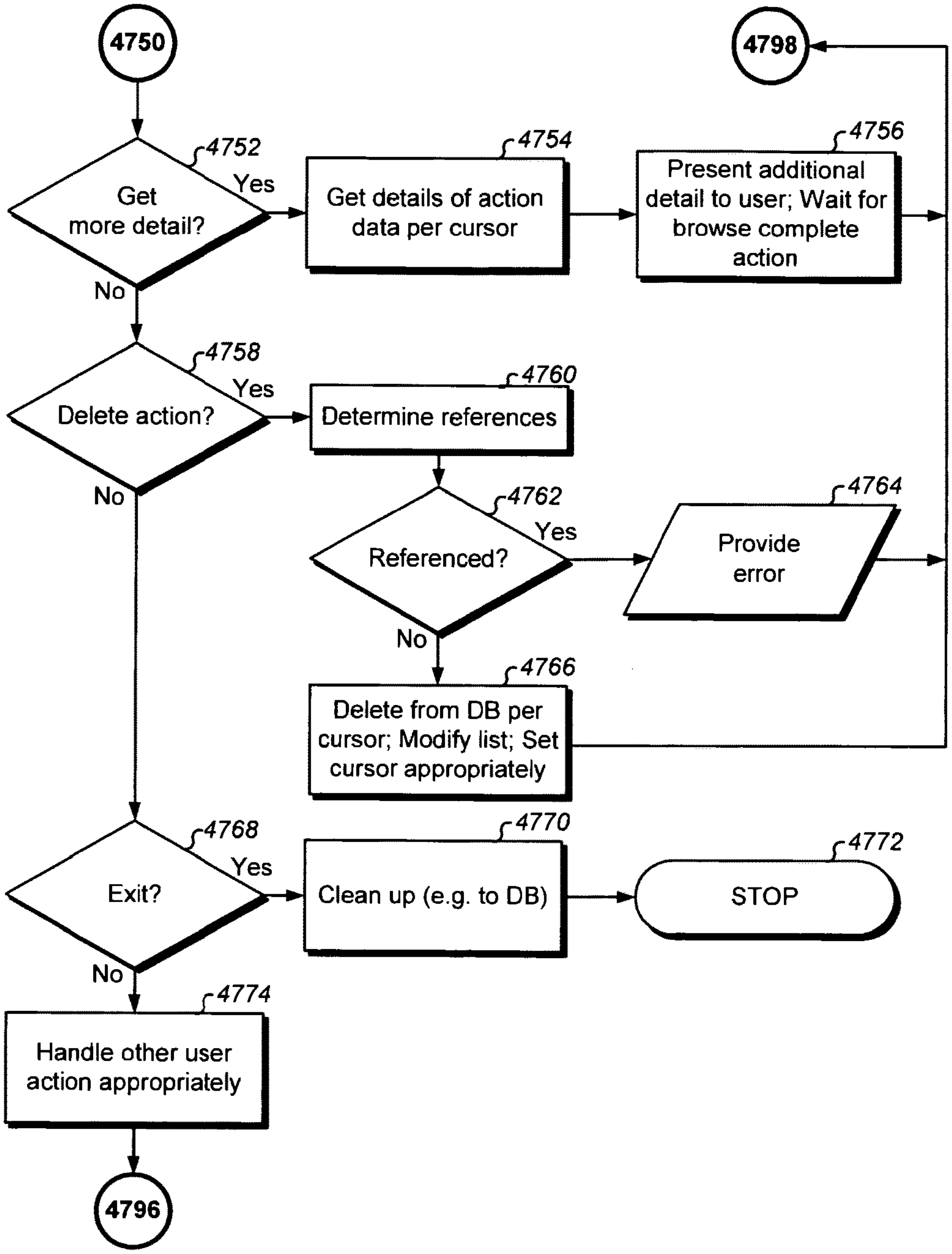

4750
4752
Get more detail?
Yes
No
4754
Get details of action data per cursor
4756
Present additional detail to user; Wait for browse complete action
4798
4758
Delete action?
Yes
No
4760
Determine references
4762
Referenced?
Yes
No
4764
Provide error
4766
Delete from DB per cursor; Modify list; Set cursor appropriately
4768
Exit?
Yes
No
4770
Clean up (e.g. to DB)
4772
STOP
4774
Handle other user action appropriately
4796

Fig. 47B

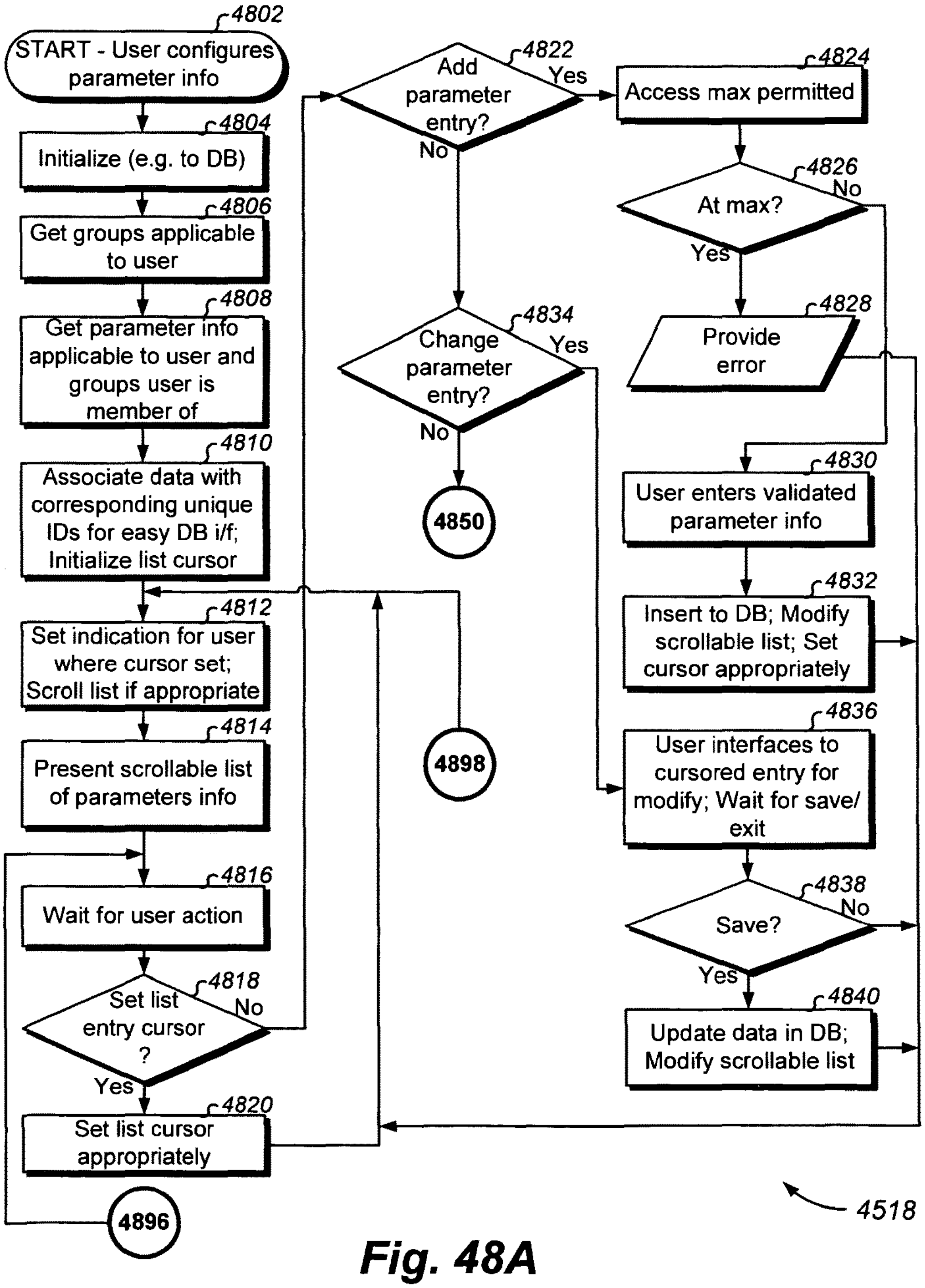

4802
START - User configures parameter info
4804
Initialize (e.g. to DB)
4806
Get groups applicable to user
4808
Get parameter info applicable to user and groups user is member of
4810
Associate data with corresponding unique IDs for easy DB i/f; Initialize list cursor
4812
Set indication for user where cursor set; Scroll list if appropriate
4814
Present scrollable list of parameters info
4816
Wait for user action
4818
Set list entry cursor ?
No
Yes
4820
Set list cursor appropriately
4896
4822
Add parameter entry?
Yes
No
4834
Change parameter entry?
Yes
No
4850
4898
4824
Access max permitted
4826
At max?
No
Yes
4828
Provide error
4830
User enters validated parameter info
4832
Insert to DB; Modify scrollable list; Set cursor appropriately
4836
User interfaces to cursored entry for modify; Wait for save/ exit
4838
Save?
No
Yes
4840
Update data in DB; Modify scrollable list
4518

Fig. 48A

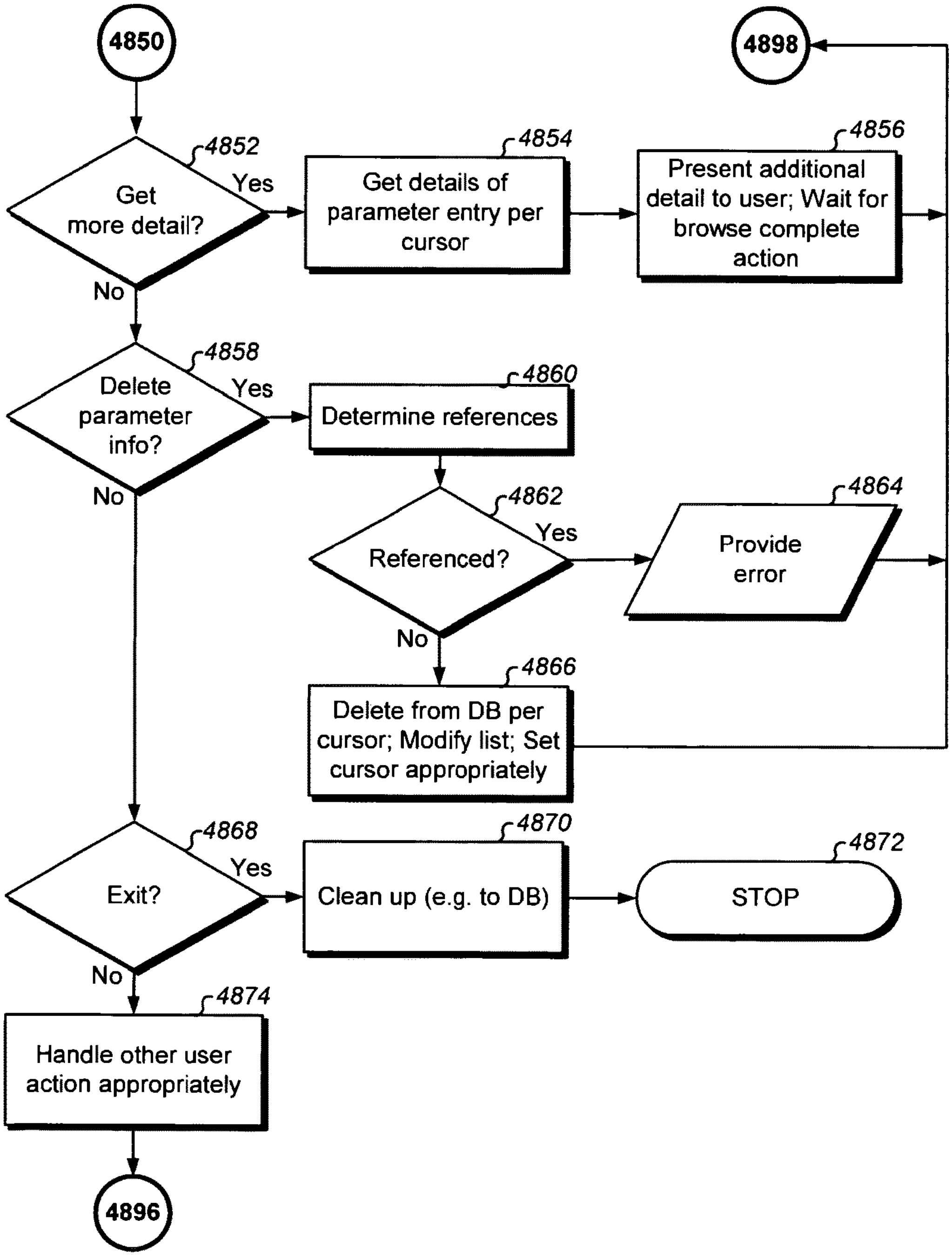
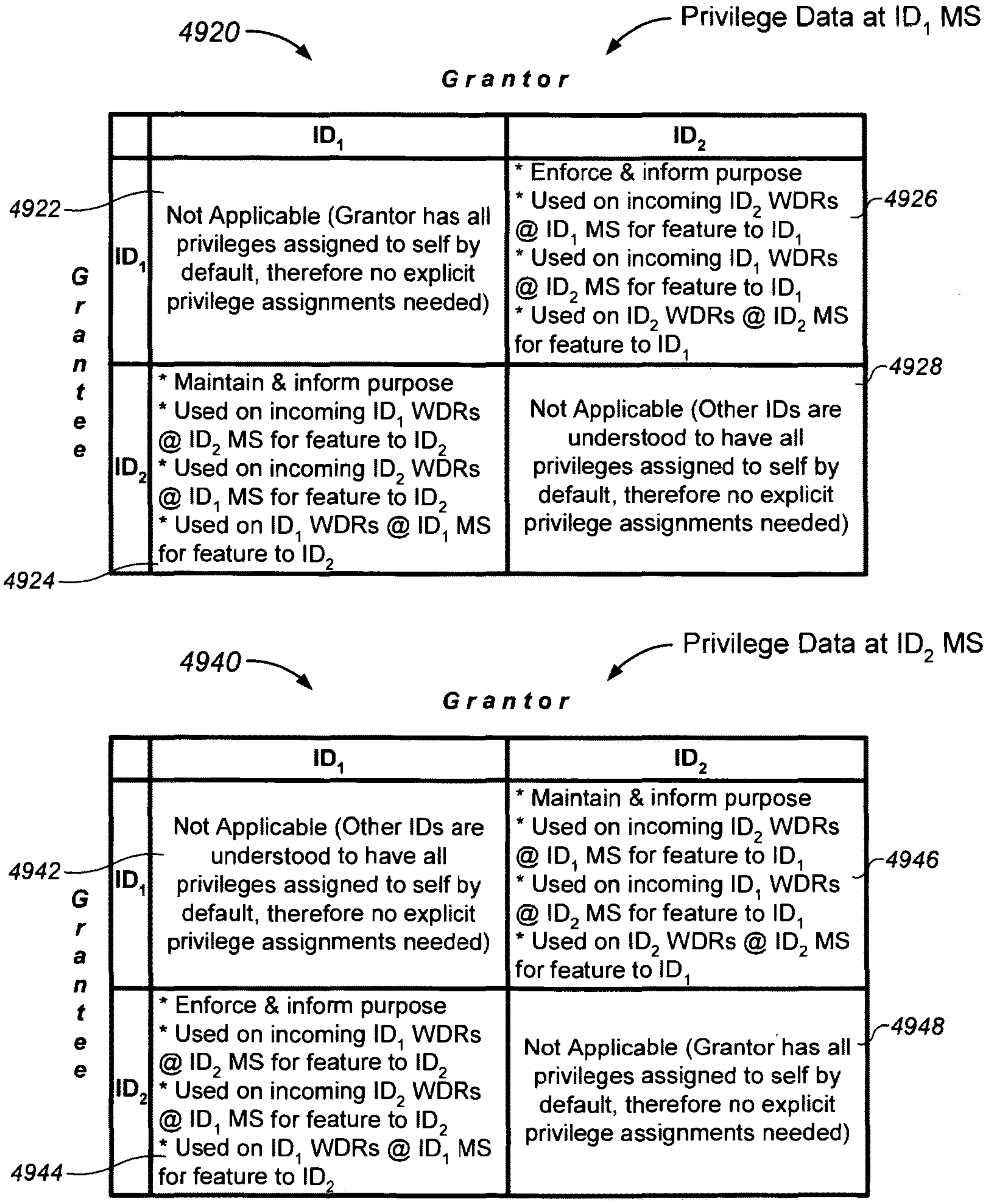

4850
4852
Get more detail?
Yes
No
4854
Get details of parameter entry per cursor
4856
Present additional detail to user; Wait for browse complete action
4898
4858
Delete parameter info?
Yes
No
4860
Determine references
4862
Referenced?
Yes
No
4864
Provide error
4866
Delete from DB per cursor; Modify list; Set cursor appropriately
4868
Exit?
Yes
No
4870
Clean up (e.g. to DB)
4872
STOP
4874
Handle other user action appropriately
4896

Fig. 48B


4920
4922
4924
4926
4928
4940
4942
4944
4946
4948

Privilege Data at ID_1 MS

Grantor

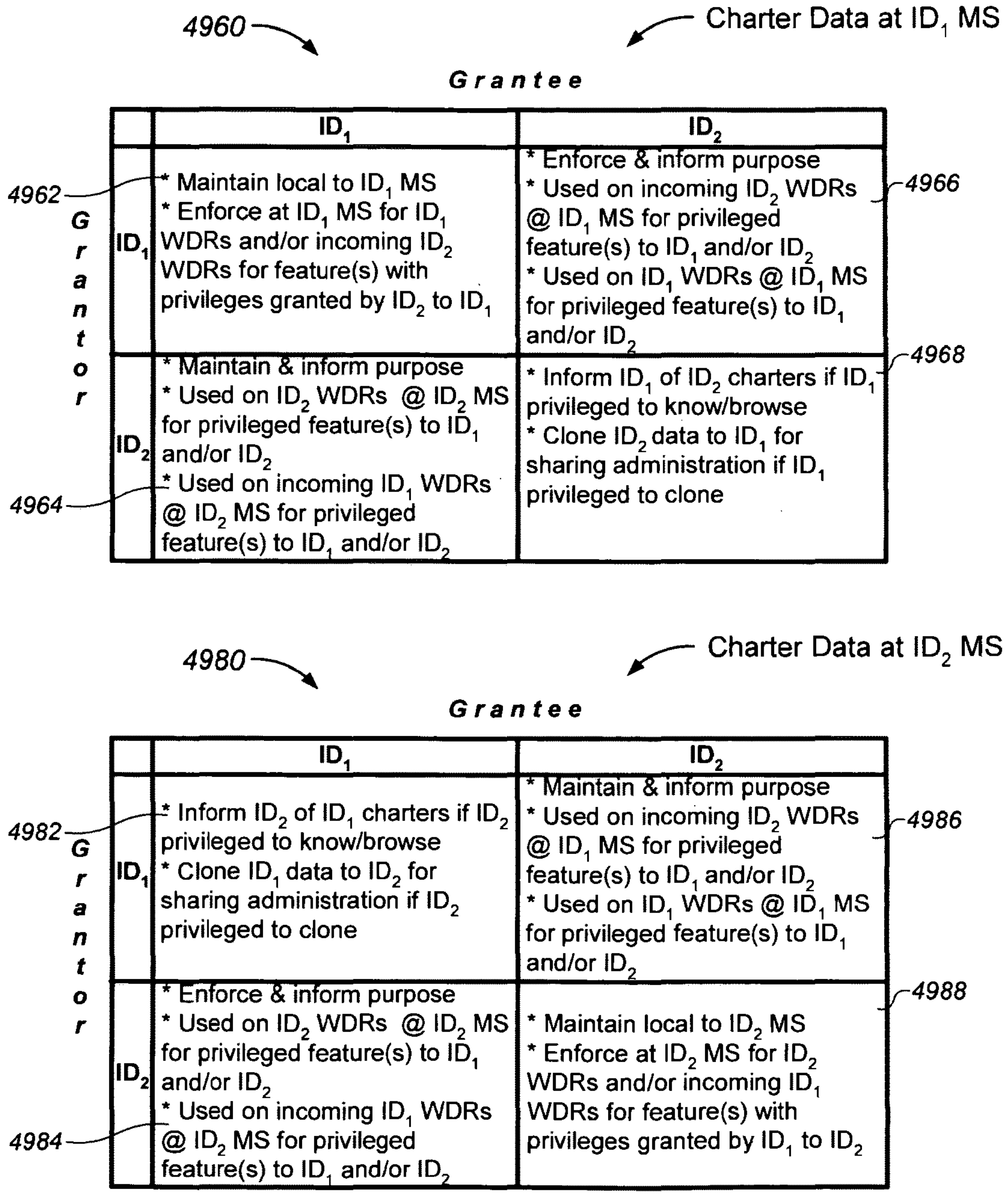
Grantee	ID_1	ID_2
ID_1	Not Applicable (Grantor has all privileges assigned to self by default, therefore no explicit privilege assignments needed)	* Enforce & inform purpose * Used on incoming ID_2 WDRs @ ID_1 MS for feature to ID_1 * Used on incoming ID_1 WDRs @ ID_2 MS for feature to ID_1 * Used on ID_2 WDRs @ ID_2 MS for feature to ID_1
ID_2	* Maintain & inform purpose * Used on incoming ID_1 WDRs @ ID_2 MS for feature to ID_2 * Used on incoming ID_2 WDRs @ ID_1 MS for feature to ID_2 * Used on ID_1 WDRs @ ID_1 MS for feature to ID_2	Not Applicable (Other IDs are understood to have all privileges assigned to self by default, therefore no explicit privilege assignments needed)

Privilege Data at ID_2 MS

Grantor

Grantee	ID_1	ID_2
ID_1	Not Applicable (Other IDs are understood to have all privileges assigned to self by default, therefore no explicit privilege assignments needed)	* Maintain & inform purpose * Used on incoming ID_2 WDRs @ ID_1 MS for feature to ID_1 * Used on incoming ID_1 WDRs @ ID_2 MS for feature to ID_1 * Used on ID_2 WDRs @ ID_2 MS for feature to ID_1
ID_2	* Enforce & inform purpose * Used on incoming ID_1 WDRs @ ID_2 MS for feature to ID_2 * Used on incoming ID_2 WDRs @ ID_1 MS for feature to ID_2 * Used on ID_1 WDRs @ ID_1 MS for feature to ID_2	Not Applicable (Grantor has all privileges assigned to self by default, therefore no explicit privilege assignments needed)

Fig. 49A


4960
4962
4964
4966
4968
4980
4982
4984
4986
4988

Charter Data at ID_1 MS

		Grantee	
		ID_1	ID_2
Grantor	ID_1	* Maintain local to ID_1 MS * Enforce at ID_1 MS for ID_1 WDRs and/or incoming ID_2 WDRs for feature(s) with privileges granted by ID_2 to ID_1	* Enforce & inform purpose * Used on incoming ID_2 WDRs @ ID_1 MS for privileged feature(s) to ID_1 and/or ID_2 * Used on ID_1 WDRs @ ID_1 MS for privileged feature(s) to ID_1 and/or ID_2
	ID_2	* Maintain & inform purpose * Used on ID_2 WDRs @ ID_2 MS for privileged feature(s) to ID_1 and/or ID_2 * Used on incoming ID_1 WDRs @ ID_2 MS for privileged feature(s) to ID_1 and/or ID_2	* Inform ID_1 of ID_2 charters if ID_1 privileged to know/browse * Clone ID_2 data to ID_1 for sharing administration if ID_1 privileged to clone

Charter Data at ID_2 MS

		Grantee	
		ID_1	ID_2
Grantor	ID_1	* Inform ID_2 of ID_1 charters if ID_2 privileged to know/browse * Clone ID_1 data to ID_2 for sharing administration if ID_2 privileged to clone	* Maintain & inform purpose * Used on incoming ID_2 WDRs @ ID_1 MS for privileged feature(s) to ID_1 and/or ID_2 * Used on ID_1 WDRs @ ID_1 MS for privileged feature(s) to ID_1 and/or ID_2
	ID_2	* Enforce & inform purpose * Used on ID_2 WDRs @ ID_2 MS for privileged feature(s) to ID_1 and/or ID_2 * Used on incoming ID_1 WDRs @ ID_2 MS for privileged feature(s) to ID_1 and/or ID_2	* Maintain local to ID_2 MS * Enforce at ID_2 MS for ID_2 WDRs and/or incoming ID_1 WDRs for feature(s) with privileges granted by ID_1 to ID_2

Fig. 49B

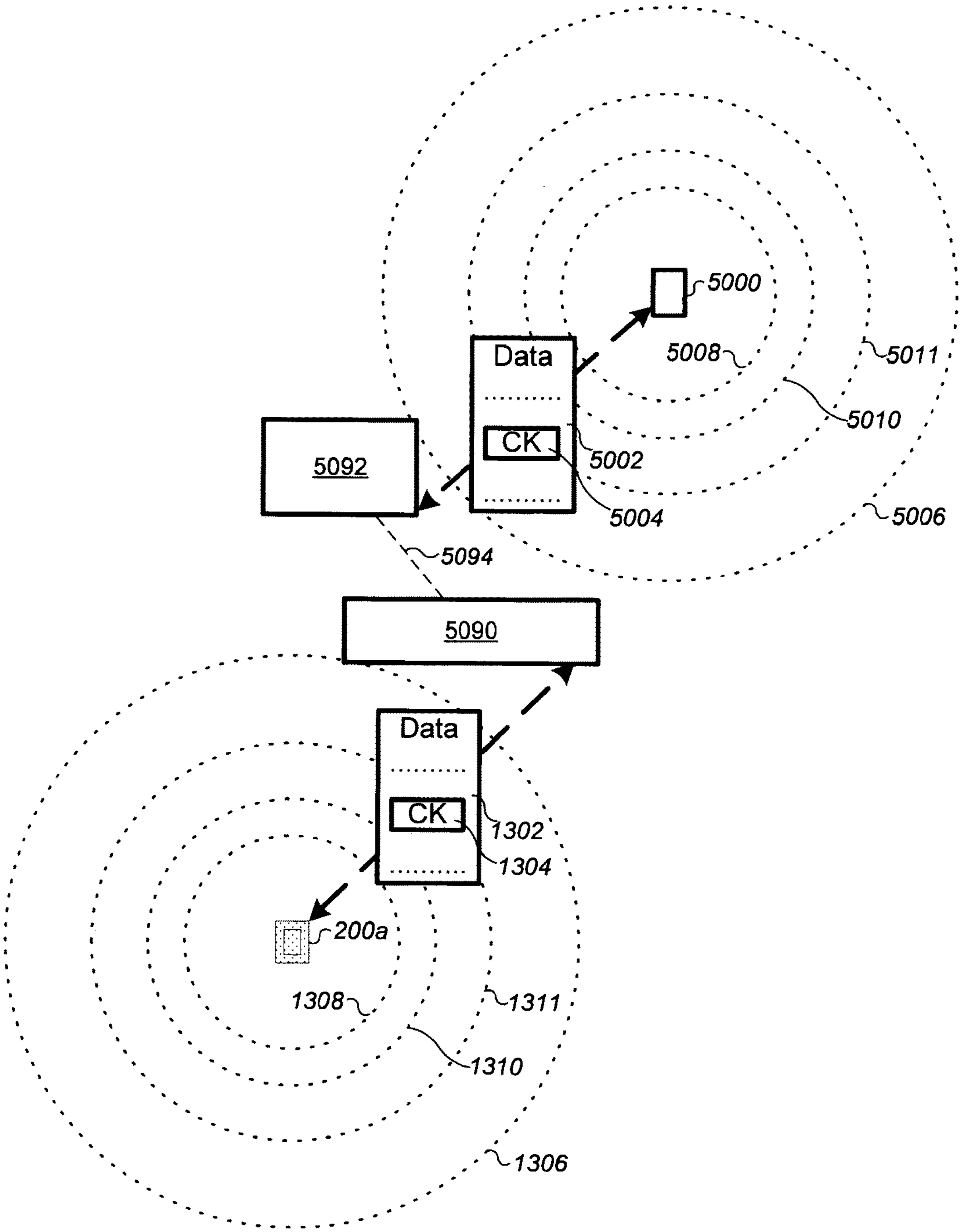

5000
Data
5008
5011
5010
CK
5002
5092
5004
5006
5094
5090
Data
CK
1302
1304
200a
1308
1311
1310
1306

Fig. 50A

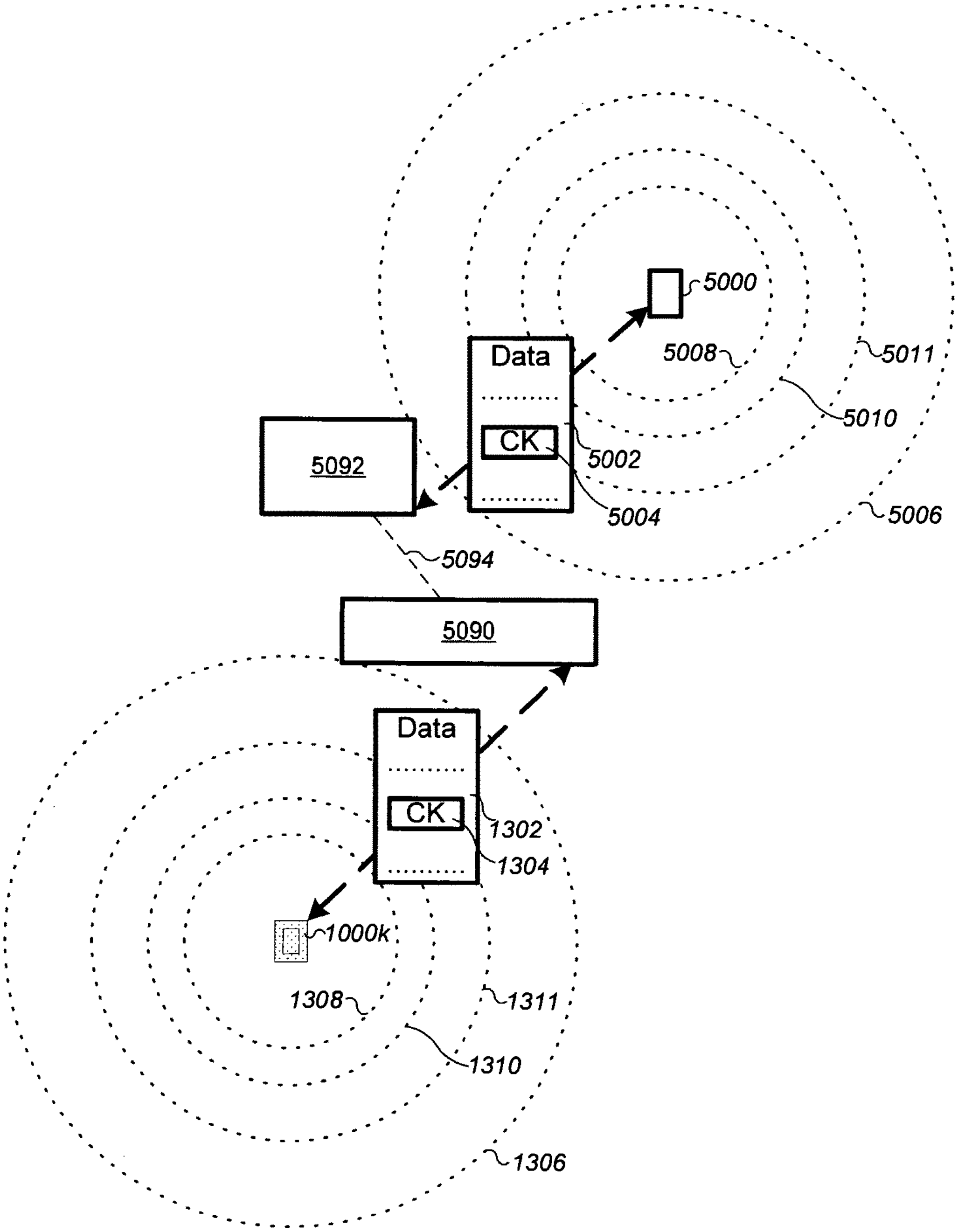

5000
Data
CK
5008
5011
5010
5002
5004
5006
5092
5094
5090
Data
CK
1302
1304
1000k
1308
1311
1310
1306

Fig. 50B

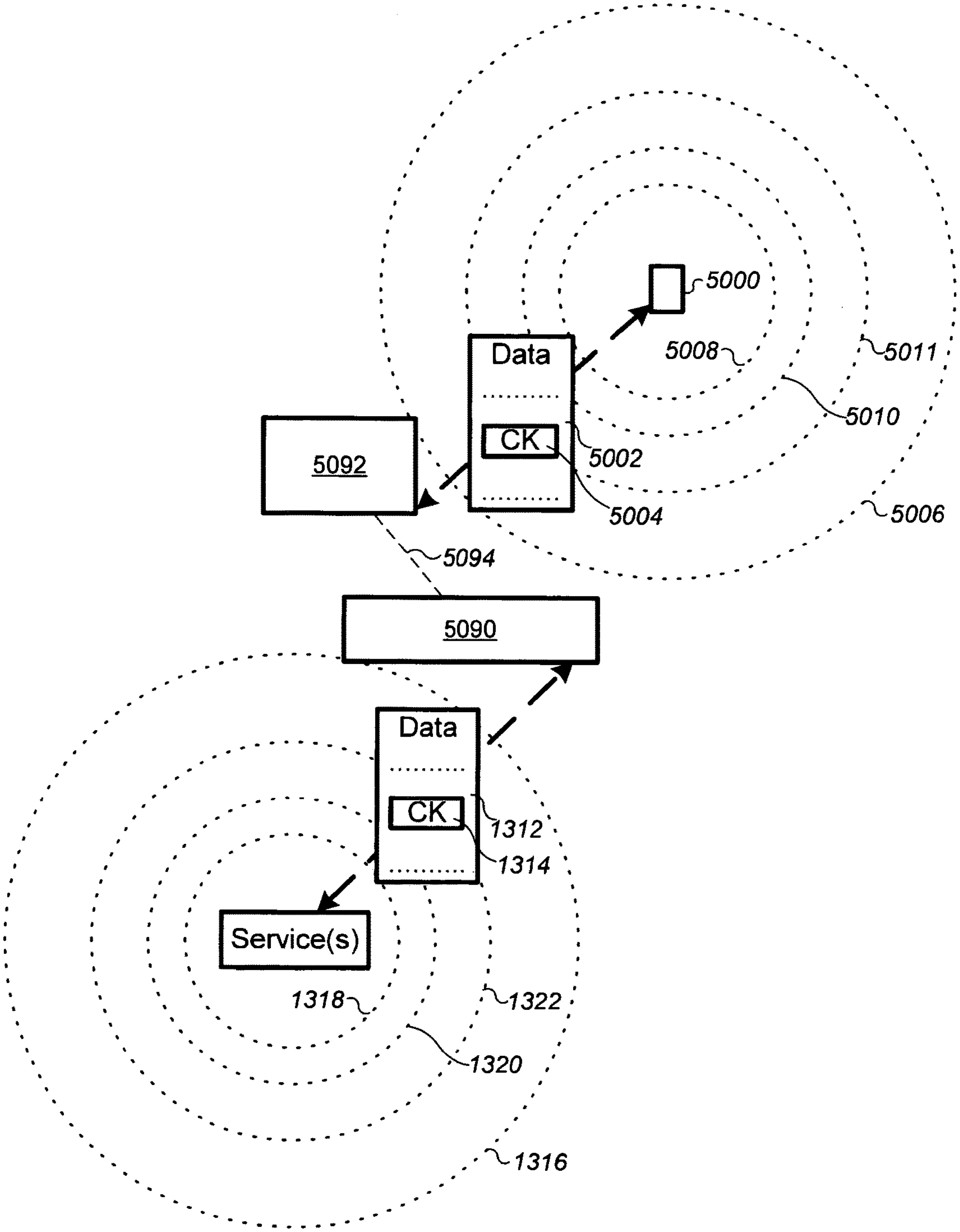
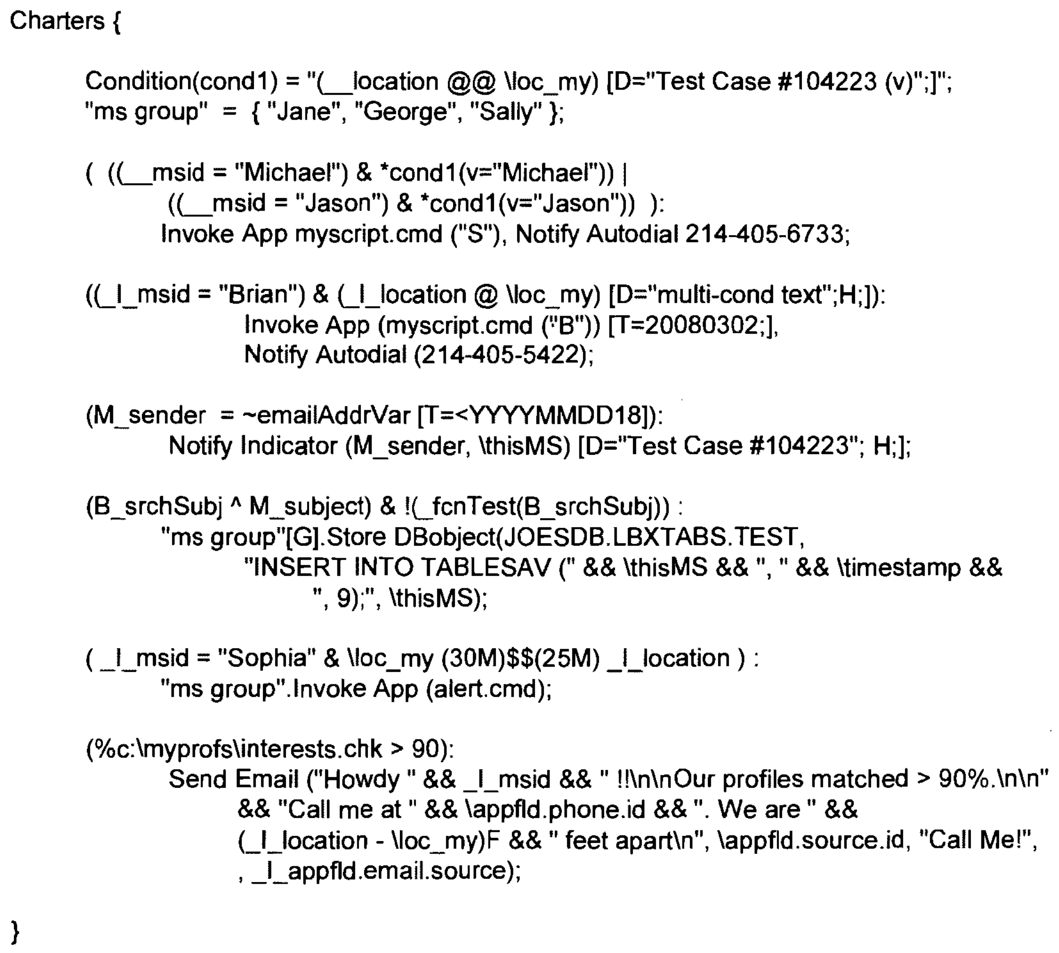

5000
Data
CK
5008
5011
5010
5002
5004
5006
5092
5094
5090
Data
CK
1312
1314
Service(s)
1318
1322
1320
1316

Fig. 50C

```
Permissions {

        Text(str) = "Test Case #106729 (context)";
        Generic(assignPrivs) = "G=Family,Work,\vuloc [T=>20080402000130.24,<20080428;
                         D=*str; H;]";
        Groups {
                LBXPHONE_USERS = Austin, Davood, Jane, Kris, Mark, Ravi, Sam, Tim;
                "SW Components" = "SM 1.0", "PIP 1.0", "PIPGUI 1.0", "SMGUI 1.0",
                                    "COMM 1.0", "KERNEL 1.1";
        }

        Grants /* Can define Grant structure(s) prior to assignment */ {
                Family= \lbxall[R=0xFFFFFFFF;] [D=*str(context="Family")];
                Work = [T=YYYYMMDD08:YYYYMMDD17;D=*str(context="Work");H;] {
                        "Department 232"=\geoar,\geode,\nearar,\nearde;
                        "Department 458" = [D="Davood Iyadi's mgt scope";] {
                                "Server Development Team" = ;
                                "lbxPhone Development Team" = {
                                        "Comm Layer Guys" = \mssys;\msbios;
                                        "GUI girls" = \msguiload;
                                        "Mark and Tim" = \msapps;
                                };
                        };
                        "Accounting Department" [H;] = \track;
                };
                Parents = { Mom=\lbxall; Dad=\lbxall; };
                Michael-Friends=\geoarr;\geode;
                Jason-Friends=\nearar;\nearde;
        }

        // Permissions are granted here:
        Bill: LBXPHONE_USERS [G=\caller;\callee;\trkall;];
        LBXPHONE_USERS: Bill [G=\callee;\caller;];
        Bill:Sophia;
        Bill:Brian [*assignPrivs];
        Bill:George [G=\geoall,\nearall;];
        Michael : Bill [G=Parents,Michael-Friends;];
        Jason: Bill [G=Parents,Jason-Friends;];

}
```

Fig. 51A



```
Charters {

        Condition(cond1) = "(__location @@ \loc_my) [D="Test Case #104223 (v)";]";
        "ms group"  =  { "Jane", "George", "Sally" };

        (  ((__msid = "Michael") & *cond1(v="Michael")) |
                ((__msid = "Jason") & *cond1(v="Jason"))  ):
               Invoke App myscript.cmd ("S"), Notify Autodial 214-405-6733;

        ((_I_msid = "Brian") & (_I_location @ \loc_my) [D="multi-cond text";H;]):
                          Invoke App (myscript.cmd ("B")) [T=20080302;],
                          Notify Autodial (214-405-5422);

        (M_sender  = ~emailAddrVar [T=<YYYYMMDD18]):
                Notify Indicator (M_sender, \thisMS) [D="Test Case #104223"; H;];

        (B_srchSubj ^ M_subject) & !(_fcnTest(B_srchSubj)) :
               "ms group"[G].Store DBobject(JOESDB.LBXTABS.TEST,
                          "INSERT INTO TABLESAV (" && \thisMS && ", " && \timestamp &&
                                 ", 9);", \thisMS);

        ( _I_msid = "Sophia" & \loc_my (30M)$$(25M) _I_location ) :
               "ms group".Invoke App (alert.cmd);

        (%c:\myprofs\interests.chk > 90):
                Send Email ("Howdy " && _I_msid && " !!\n\nOur profiles matched > 90%.\n\n"
                       && "Call me at " && \appfld.phone.id && ". We are " &&
                       (_I_location - \loc_my)F && " feet apart\n", \appfld.source.id, "Call Me!",
                       , _I_appfld.email.source);

}
```

Fig. 51B

```
typedef struct privilege {
            unsigned long         priv;
            unsigned char         relevance[MAX_RELEVANCEMASK];
            TIMESPEC              *tspec;       // merged with permission level (if permission
                                                // level was present)
            struct privilege      *nextPriv;
            } PRIVILEGE;

typedef struct permission {
            unsigned char         grantor[MAX_IDLENGTH];
            unsigned short        grantor_idtype;
            unsigned char         grantee[MAX_IDLENGTH];
            unsigned short        grantee_idtype;
            PRIVILEGE             *privileges;
            struct permission     *nextPerm;
            } PERMISSION;

typedef struct action {
            IDENTITY              host;
            unsigned short        cmd;
            unsigned short        operand;
            unsigned char         *params;
            TIMESPEC              *tspec;             // merged with charter level (if charter
                                                      // level was present)
            struct action         *nextActn;
            } ACTION;

typedef struct charter {
            unsigned char         grantee[MAX_IDLENGTH];
            unsigned short        grantee_idtype;
            unsigned char         grantor[MAX_IDLENGTH];
            unsigned short        grantor_idtype;
            unsigned char         *expression;
            ACTION                *actn;
            struct charter        *nextCharter;
            } CHARTER;
```

Fig. 52

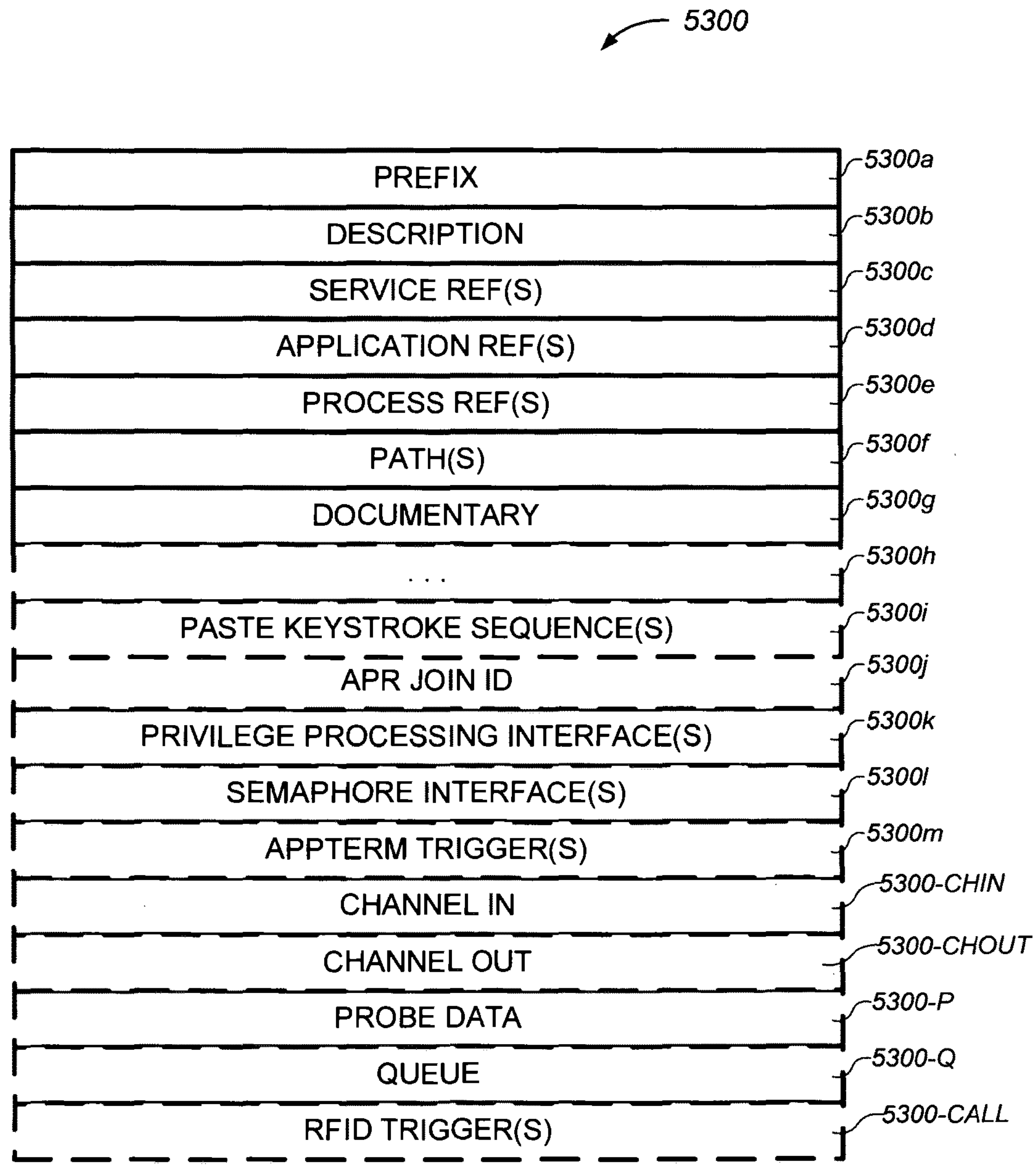

5300
PREFIX
5300a
DESCRIPTION
5300b
SERVICE REF(S)
5300c
APPLICATION REF(S)
5300d
PROCESS REF(S)
5300e
PATH(S)
5300f
DOCUMENTARY
5300g
. . .
5300h
PASTE KEYSTROKE SEQUENCE(S)
5300i
APR JOIN ID
5300j
PRIVILEGE PROCESSING INTERFACE(S)
5300k
SEMAPHORE INTERFACE(S)
5300l
APPTERM TRIGGER(S)
5300m
CHANNEL IN
5300-CHIN
CHANNEL OUT
5300-CHOUT
PROBE DATA
5300-P
QUEUE
5300-Q
RFID TRIGGER(S)
5300-CALL

Fig. 53

```
...
x = "this is a textual description"
...
z="timespec=""200802030000:200812312359"" description=""test98341;Permission"""
...
<permission grantor="Jimbob" grantee="Henry" <%=z%> >
        <grant name="grant1" >
                <privilege id="\lbxcpy" relevance="FFFFFFFF"
                        timespec="YYYMMDD09:YYMMDD17" description="<%=x%>" />
                <privilege id="\lbxflt" />
                ...
        </grant>
        ...
</permission>
...
<group name="group123" >
        <member="Jim" />
        <member="Sue" />
        ...
</group>
...
<charter grantee="Henry" grantor="Jimbob" timespec="200802030000:200812312359"
        description="test98341;Charter" >
        <expression>
                <condition trigger="true"
                        specification="(__msid = ""Michael"") & __location $(300M) \loc_my" />
                <condition trigger="true"
                        specification="(__msid = ""Jason"") & __location $(300M) \loc_my" />
        </expression>
        <action host="George" cmd="Invoke" operand="App" param="alert.cmd" />
        <action host="George" cmd="Notify" operand="Indicator" param="test98341 Fired!" />
</charter>
...
```

Fig. 54

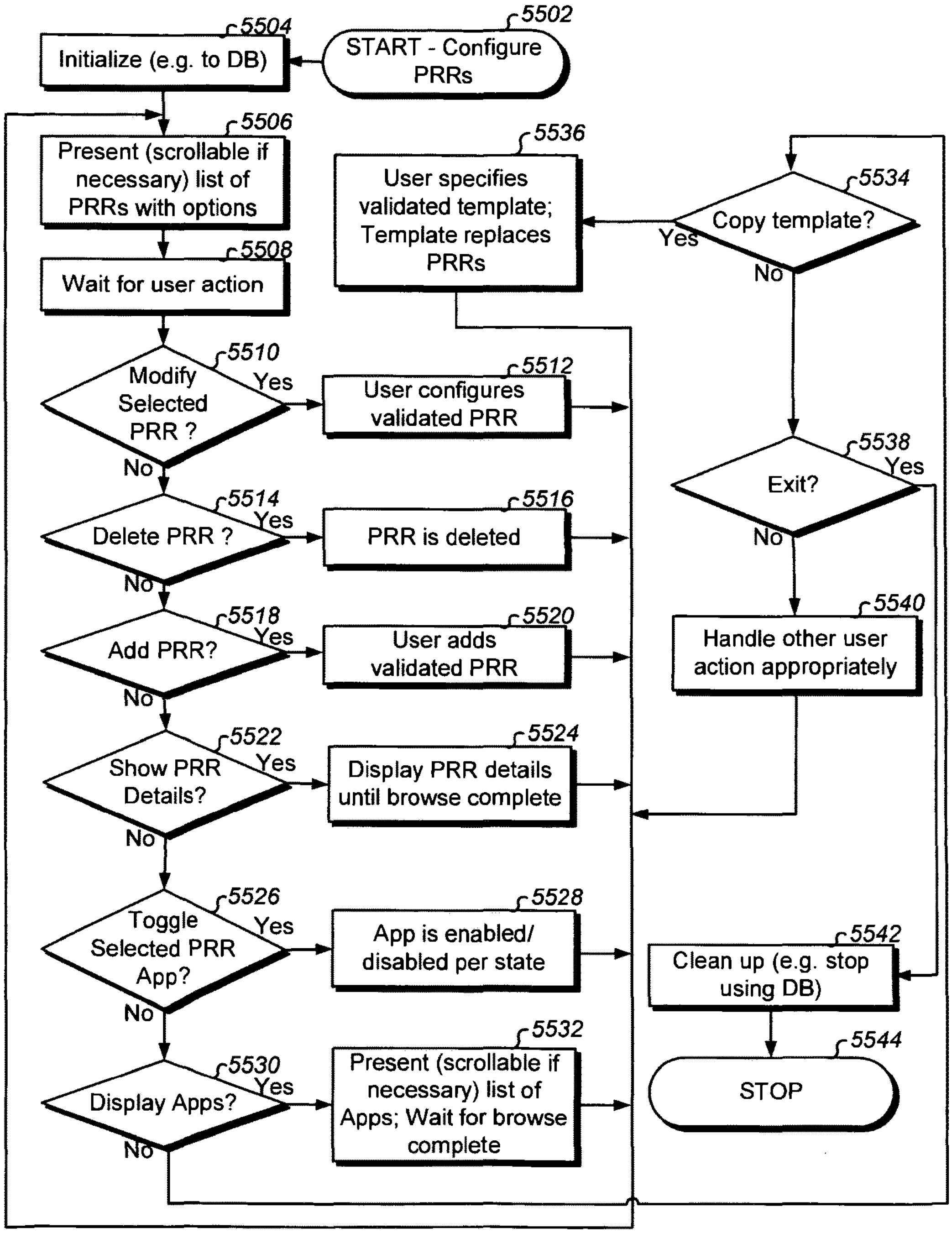

5502
START - Configure PRRs
5504
Initialize (e.g. to DB)
5506
Present (scrollable if necessary) list of PRRs with options
5508
Wait for user action
5510
Modify Selected PRR ?
Yes
No
5512
User configures validated PRR
5514
Delete PRR ?
Yes
No
5516
PRR is deleted
5518
Add PRR?
Yes
No
5520
User adds validated PRR
5522
Show PRR Details?
Yes
No
5524
Display PRR details until browse complete
5526
Toggle Selected PRR App?
Yes
No
5528
App is enabled/ disabled per state
5530
Display Apps?
Yes
No
5532
Present (scrollable if necessary) list of Apps; Wait for browse complete
5536
User specifies validated template; Template replaces PRRs
5534
Copy template?
Yes
No
5538
Exit?
Yes
No
5540
Handle other user action appropriately
5542
Clean up (e.g. stop using DB)
5544
STOP

Fig. 55A

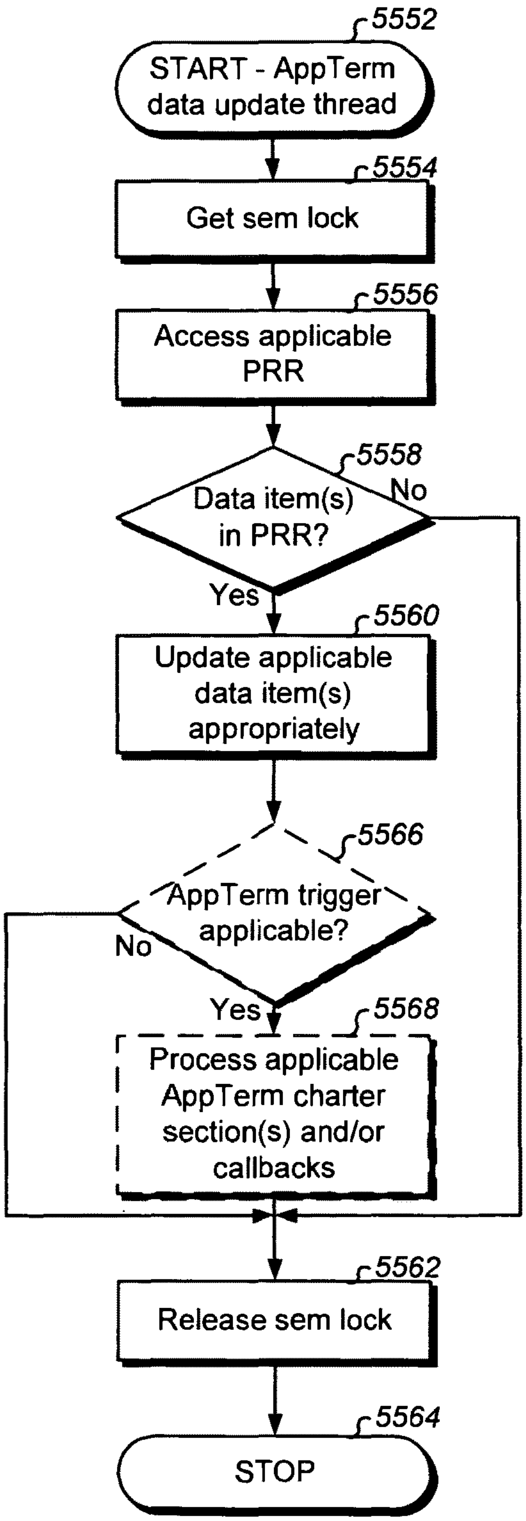

5552
START - AppTerm data update thread
5554
Get sem lock
5556
Access applicable PRR
5558
Data item(s) in PRR?
No
Yes
5560
Update applicable data item(s) appropriately
5566
AppTerm trigger applicable?
No
Yes
5568
Process applicable AppTerm charter section(s) and/or callbacks
5562
Release sem lock
5564
STOP

Fig. 55B

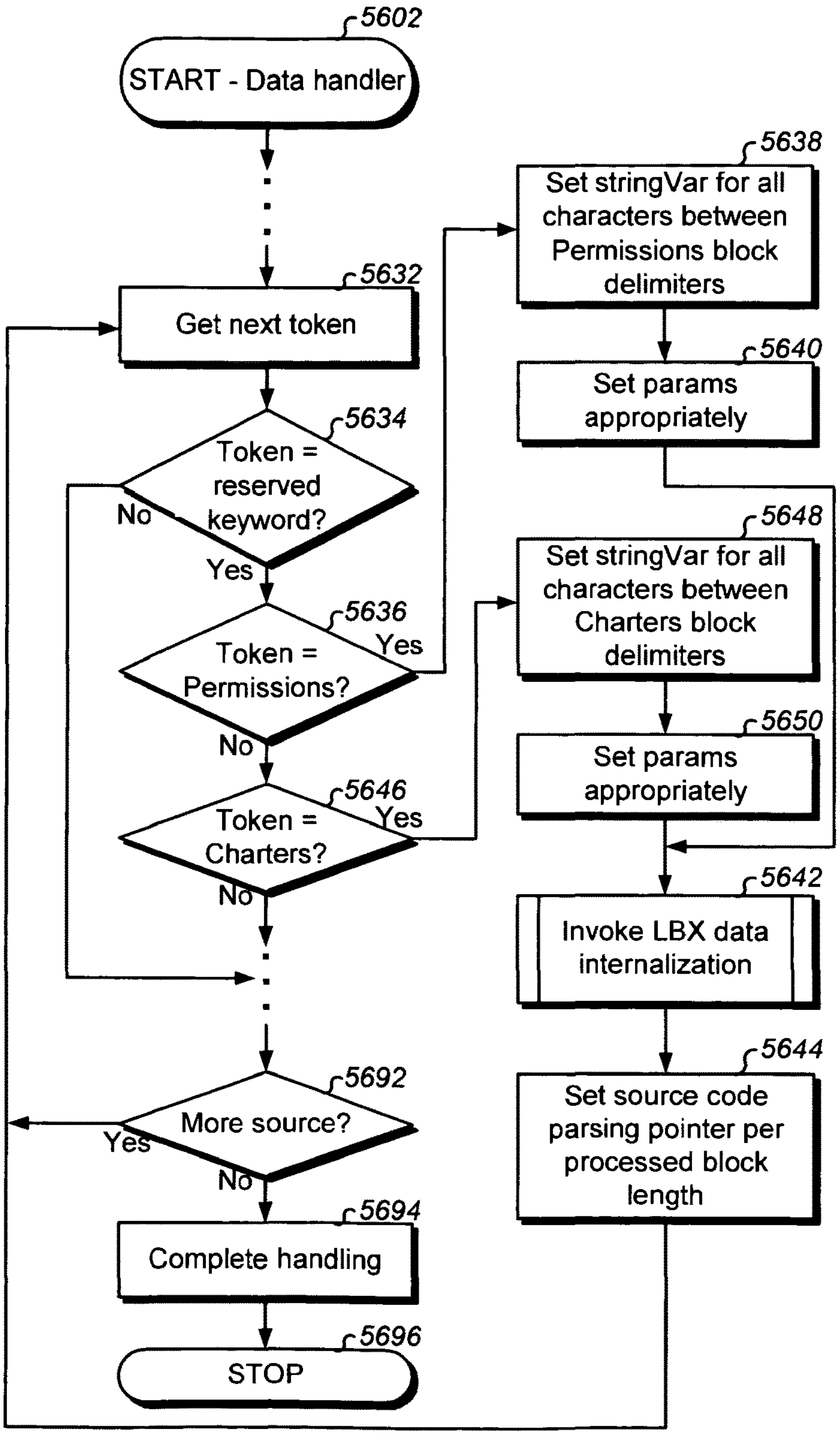

5602
START - Data handler
5632
Get next token
5634
Token = reserved keyword?
No
Yes
5636
Token = Permissions?
Yes
No
5646
Token = Charters?
Yes
No
5692
More source?
Yes
No
5694
Complete handling
5696
STOP
5638
Set stringVar for all characters between Permissions block delimiters
5640
Set params appropriately
5648
Set stringVar for all characters between Charters block delimiters
5650
Set params appropriately
5642
Invoke LBX data internalization
5644
Set source code parsing pointer per processed block length

Fig. 56

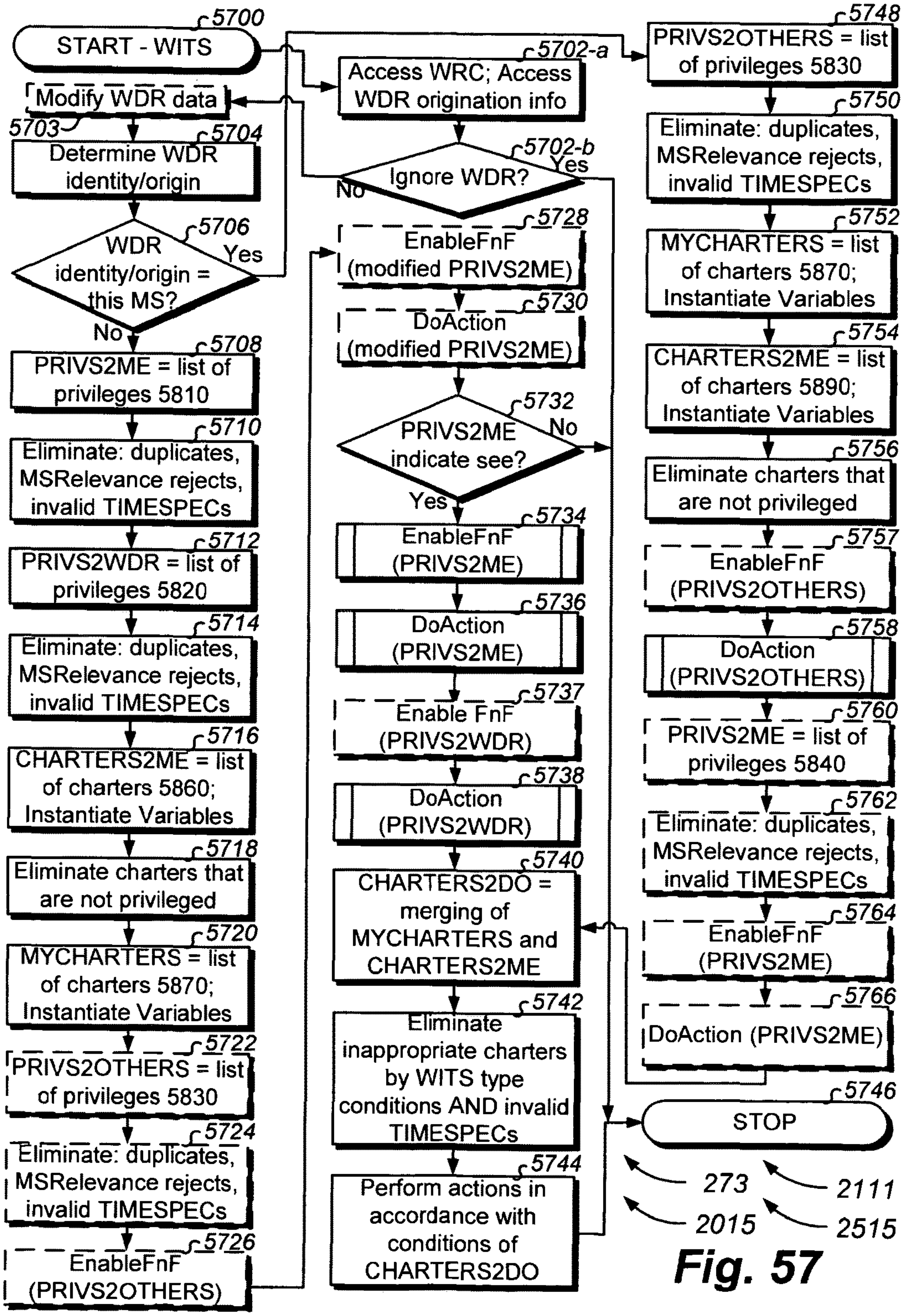

5700
START - WITS
Modify WDR data
5703
5704
Determine WDR identity/origin
5706
WDR identity/origin = this MS?
Yes
No
5708
PRIVS2ME = list of privileges 5810
5710
Eliminate: duplicates, MSRelevance rejects, invalid TIMESPECs
5712
PRIVS2WDR = list of privileges 5820
5714
Eliminate: duplicates, MSRelevance rejects, invalid TIMESPECs
5716
CHARTERS2ME = list of charters 5860; Instantiate Variables
5718
Eliminate charters that are not privileged
5720
MYCHARTERS = list of charters 5870; Instantiate Variables
5722
PRIVS2OTHERS = list of privileges 5830
5724
Eliminate: duplicates, MSRelevance rejects, invalid TIMESPECs
5726
EnableFnF (PRIVS2OTHERS)
5702-a
Access WRC; Access WDR origination info
5702-b
Ignore WDR?
Yes
No
5728
EnableFnF (modified PRIVS2ME)
5730
DoAction (modified PRIVS2ME)
5732
PRIVS2ME indicate see?
No
Yes
5734
EnableFnF (PRIVS2ME)
5736
DoAction (PRIVS2ME)
5737
Enable FnF (PRIVS2WDR)
5738
DoAction (PRIVS2WDR)
5740
CHARTERS2DO = merging of MYCHARTERS and CHARTERS2ME
5742
Eliminate inappropriate charters by WITS type conditions AND invalid TIMESPECs
5744
Perform actions in accordance with conditions of CHARTERS2DO
5748
PRIVS2OTHERS = list of privileges 5830
5750
Eliminate: duplicates, MSRelevance rejects, invalid TIMESPECs
5752
MYCHARTERS = list of charters 5870; Instantiate Variables
5754
CHARTERS2ME = list of charters 5890; Instantiate Variables
5756
Eliminate charters that are not privileged
5757
EnableFnF (PRIVS2OTHERS)
5758
DoAction (PRIVS2OTHERS)
5760
PRIVS2ME = list of privileges 5840
5762
Eliminate: duplicates, MSRelevance rejects, invalid TIMESPECs
5764
EnableFnF (PRIVS2ME)
5766
DoAction (PRIVS2ME)
5746
STOP
273
2111
2015
2515

Fig. 57

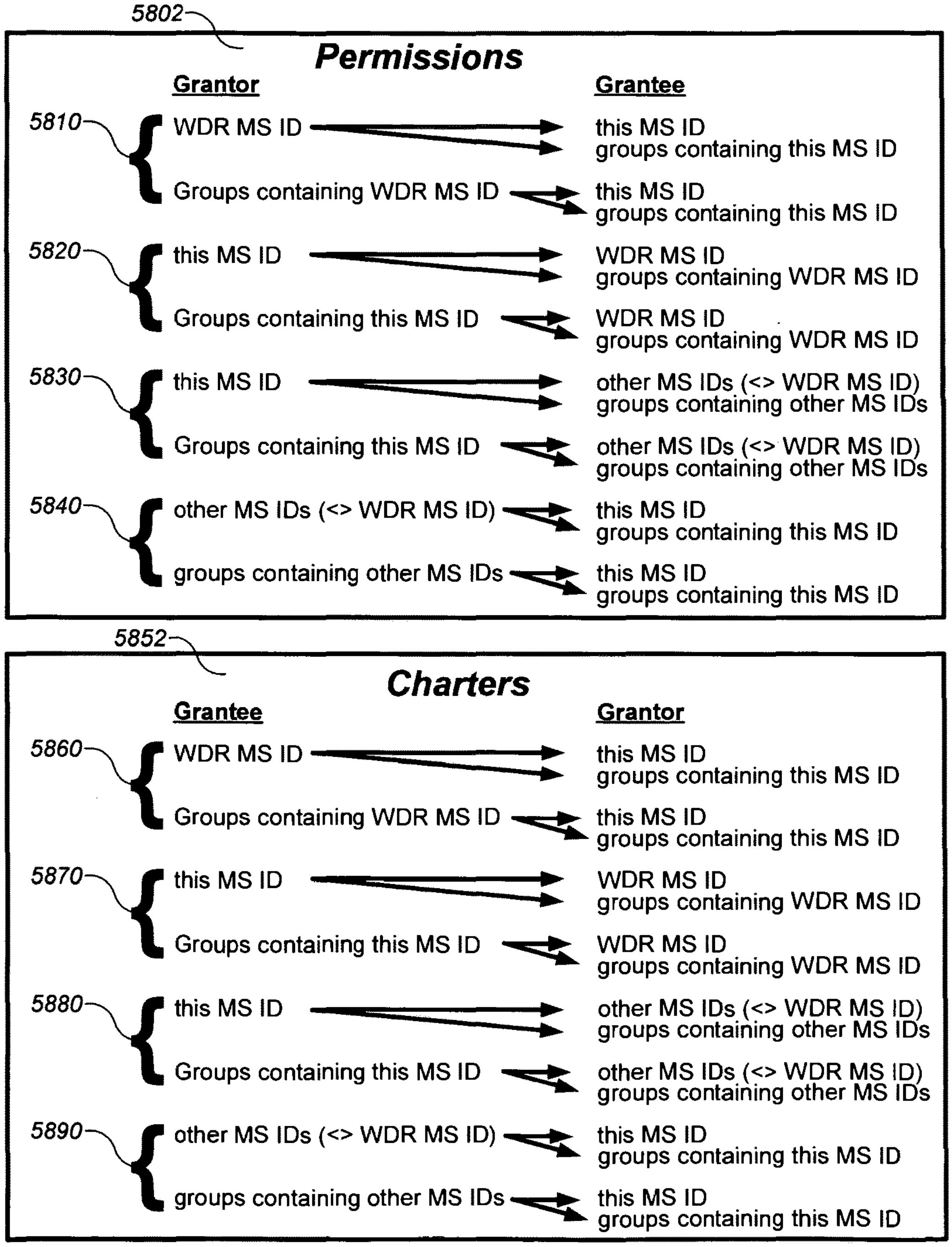

5802
Permissions
Grantor
Grantee
5810
WDR MS ID
this MS ID
groups containing this MS ID
Groups containing WDR MS ID
this MS ID
groups containing this MS ID
5820
this MS ID
WDR MS ID
groups containing WDR MS ID
Groups containing this MS ID
WDR MS ID
groups containing WDR MS ID
5830
this MS ID
other MS IDs (<> WDR MS ID)
groups containing other MS IDs
Groups containing this MS ID
other MS IDs (<> WDR MS ID)
groups containing other MS IDs
5840
other MS IDs (<> WDR MS ID)
this MS ID
groups containing this MS ID
groups containing other MS IDs
this MS ID
groups containing this MS ID
5852
Charters
Grantee
Grantor
5860
WDR MS ID
this MS ID
groups containing this MS ID
Groups containing WDR MS ID
this MS ID
groups containing this MS ID
5870
this MS ID
WDR MS ID
groups containing WDR MS ID
Groups containing this MS ID
WDR MS ID
groups containing WDR MS ID
5880
this MS ID
other MS IDs (<> WDR MS ID)
groups containing other MS IDs
Groups containing this MS ID
other MS IDs (<> WDR MS ID)
groups containing other MS IDs
5890
other MS IDs (<> WDR MS ID)
this MS ID
groups containing this MS ID
groups containing other MS IDs
this MS ID
groups containing this MS ID

Fig. 58

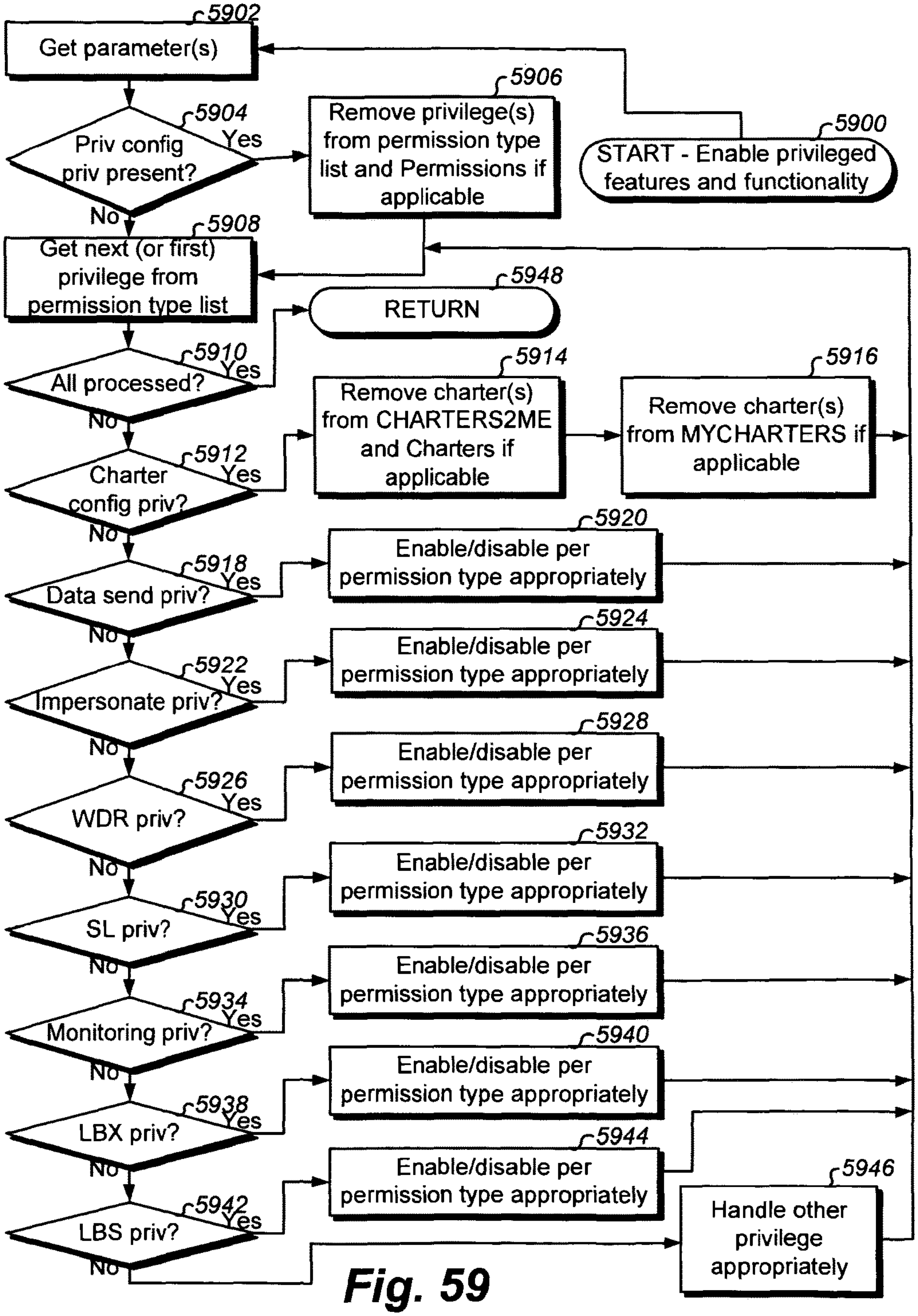

5900
START - Enable privileged features and functionality
5902
Get parameter(s)
5904
Priv config priv present?
Yes
No
5906
Remove privilege(s) from permission type list and Permissions if applicable
5908
Get next (or first) privilege from permission type list
5948
RETURN
5910
All processed?
Yes
No
5912
Charter config priv?
Yes
No
5914
Remove charter(s) from CHARTERS2ME and Charters if applicable
5916
Remove charter(s) from MYCHARTERS if applicable
5918
Data send priv?
Yes
No
5920
Enable/disable per permission type appropriately
5922
Impersonate priv?
Yes
No
5924
Enable/disable per permission type appropriately
5926
WDR priv?
Yes
No
5928
Enable/disable per permission type appropriately
5930
SL priv?
Yes
No
5932
Enable/disable per permission type appropriately
5934
Monitoring priv?
Yes
No
5936
Enable/disable per permission type appropriately
5938
LBX priv?
Yes
No
5940
Enable/disable per permission type appropriately
5942
LBS priv?
Yes
No
5944
Enable/disable per permission type appropriately
5946
Handle other privilege appropriately

Fig. 59

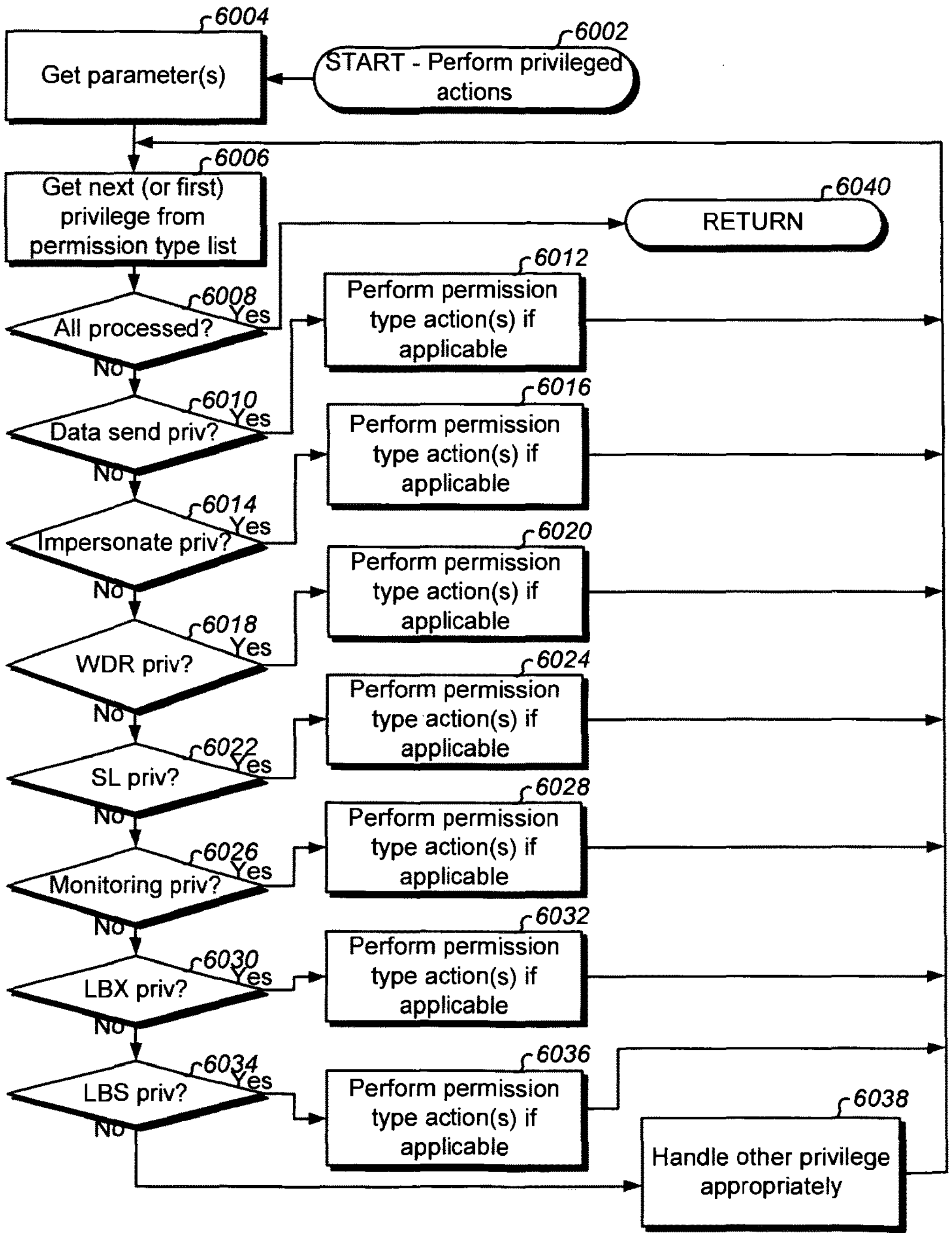
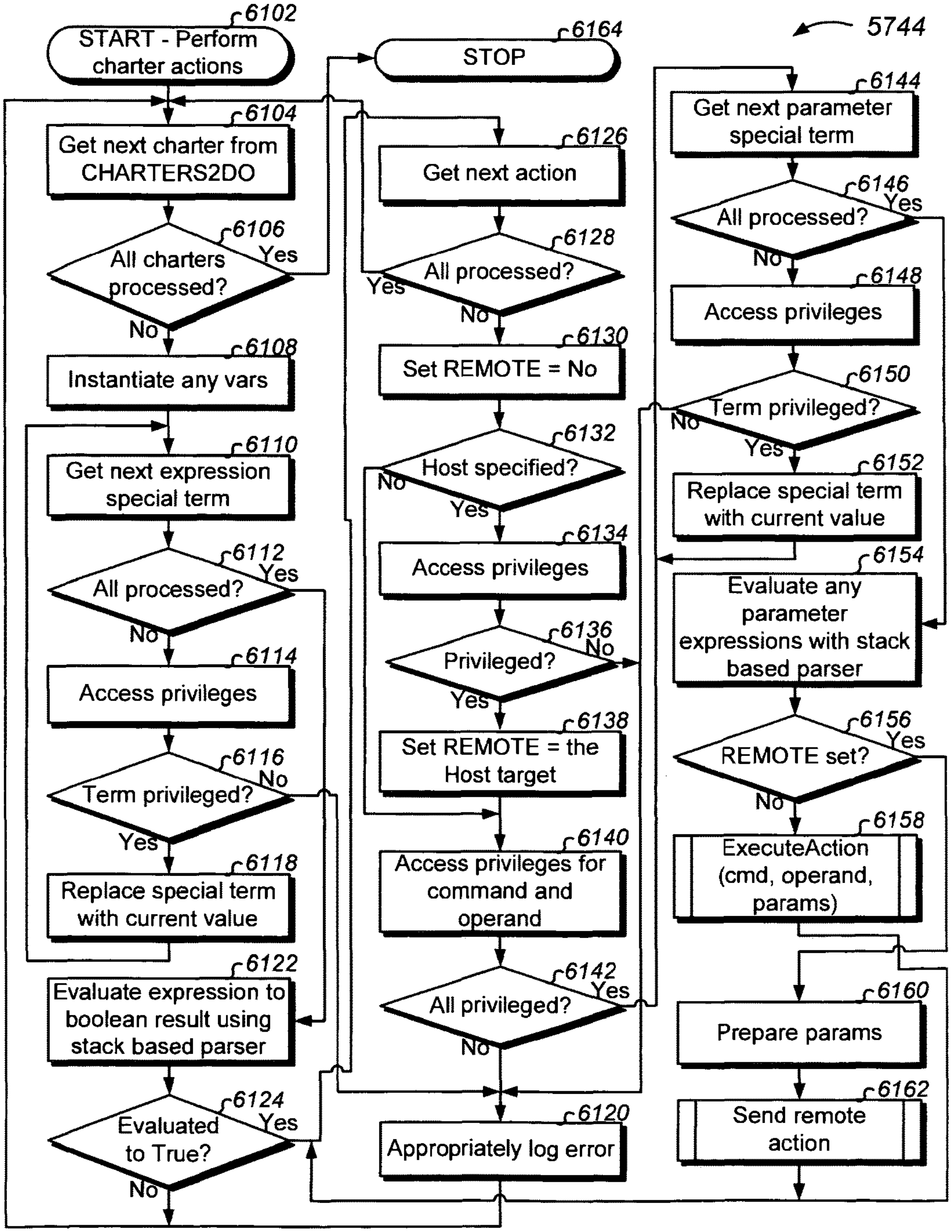

6002
START - Perform privileged actions
6004
Get parameter(s)
6006
Get next (or first) privilege from permission type list
6008
All processed?
Yes
No
6040
RETURN
6010
Data send priv?
6012
Perform permission type action(s) if applicable
6014
Impersonate priv?
6016
Perform permission type action(s) if applicable
6018
WDR priv?
6020
Perform permission type action(s) if applicable
6022
SL priv?
6024
Perform permission type action(s) if applicable
6026
Monitoring priv?
6028
Perform permission type action(s) if applicable
6030
LBX priv?
6032
Perform permission type action(s) if applicable
6034
LBS priv?
6036
Perform permission type action(s) if applicable
6038
Handle other privilege appropriately

Fig. 60



5744
6102 START - Perform charter actions
6104 Get next charter from CHARTERS2DO
6106 All charters processed?
Yes
No
6108 Instantiate any vars
6110 Get next expression special term
6112 All processed?
Yes
No
6114 Access privileges
6116 Term privileged?
No
Yes
6118 Replace special term with current value
6122 Evaluate expression to boolean result using stack based parser
6124 Evaluated to True?
Yes
No
6164 STOP
6126 Get next action
6128 All processed?
Yes
No
6130 Set REMOTE = No
6132 Host specified?
No
Yes
6134 Access privileges
6136 Privileged?
No
Yes
6138 Set REMOTE = the Host target
6140 Access privileges for command and operand
6142 All privileged?
Yes
No
6120 Appropriately log error
6144 Get next parameter special term
6146 All processed?
Yes
No
6148 Access privileges
6150 Term privileged?
No
Yes
6152 Replace special term with current value
6154 Evaluate any parameter expressions with stack based parser
6156 REMOTE set?
Yes
No
6158 ExecuteAction (cmd, operand, params)
6160 Prepare params
6162 Send remote action

Fig. 61

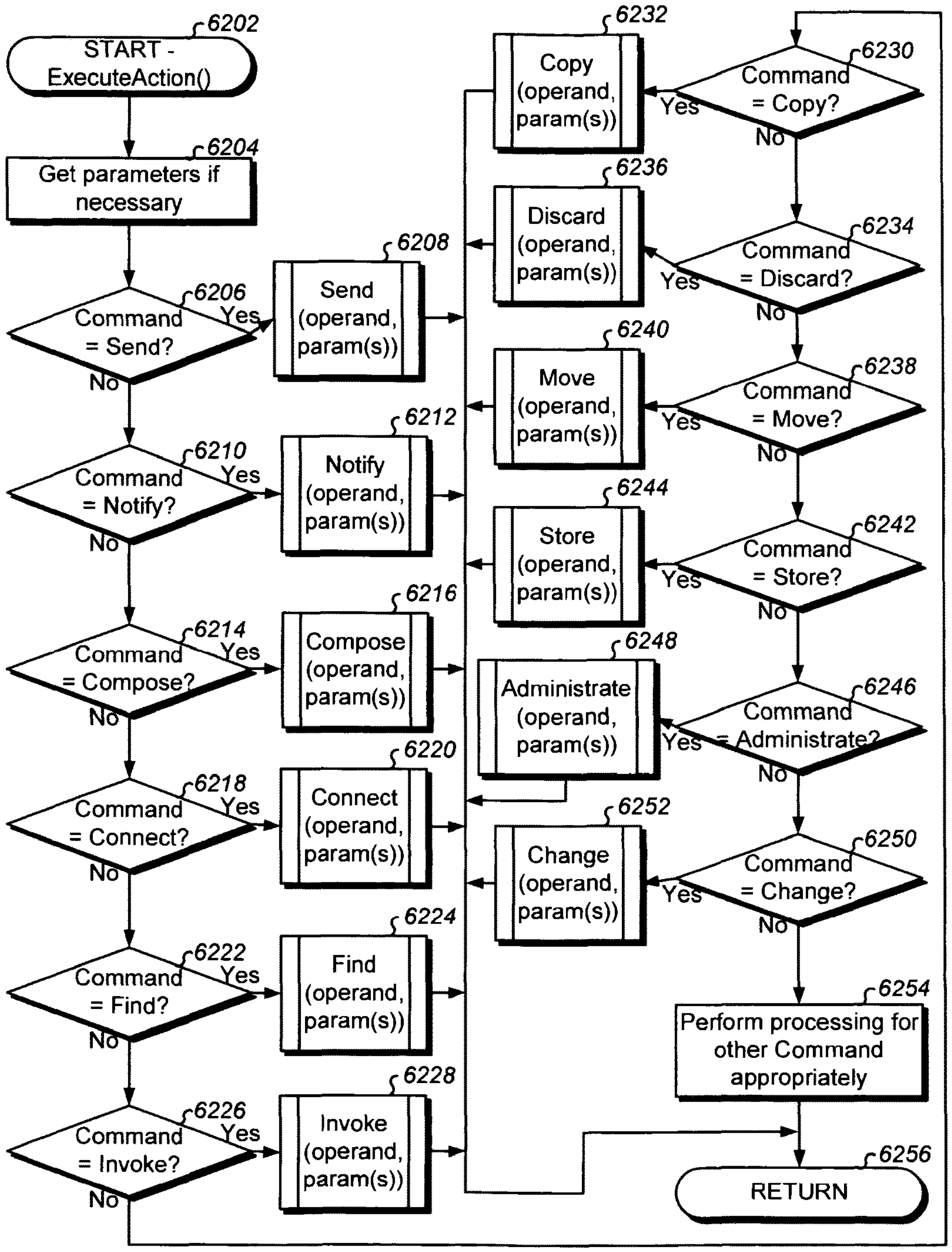

6202
START - ExecuteAction()
6204
Get parameters if necessary
6206
Command = Send?
Yes
No
6208
Send (operand, param(s))
6210
Command = Notify?
Yes
No
6212
Notify (operand, param(s))
6214
Command = Compose?
Yes
No
6216
Compose (operand, param(s))
6218
Command = Connect?
Yes
No
6220
Connect (operand, param(s))
6222
Command = Find?
Yes
No
6224
Find (operand, param(s))
6226
Command = Invoke?
Yes
No
6228
Invoke (operand, param(s))
6230
Command = Copy?
Yes
No
6232
Copy (operand, param(s))
6234
Command = Discard?
Yes
No
6236
Discard (operand, param(s))
6238
Command = Move?
Yes
No
6240
Move (operand, param(s))
6242
Command = Store?
Yes
No
6244
Store (operand, param(s))
6246
Command = Administrate?
Yes
No
6248
Administrate (operand, param(s))
6250
Command = Change?
Yes
No
6252
Change (operand, param(s))
6254
Perform processing for other Command appropriately
6256
RETURN

Fig. 62

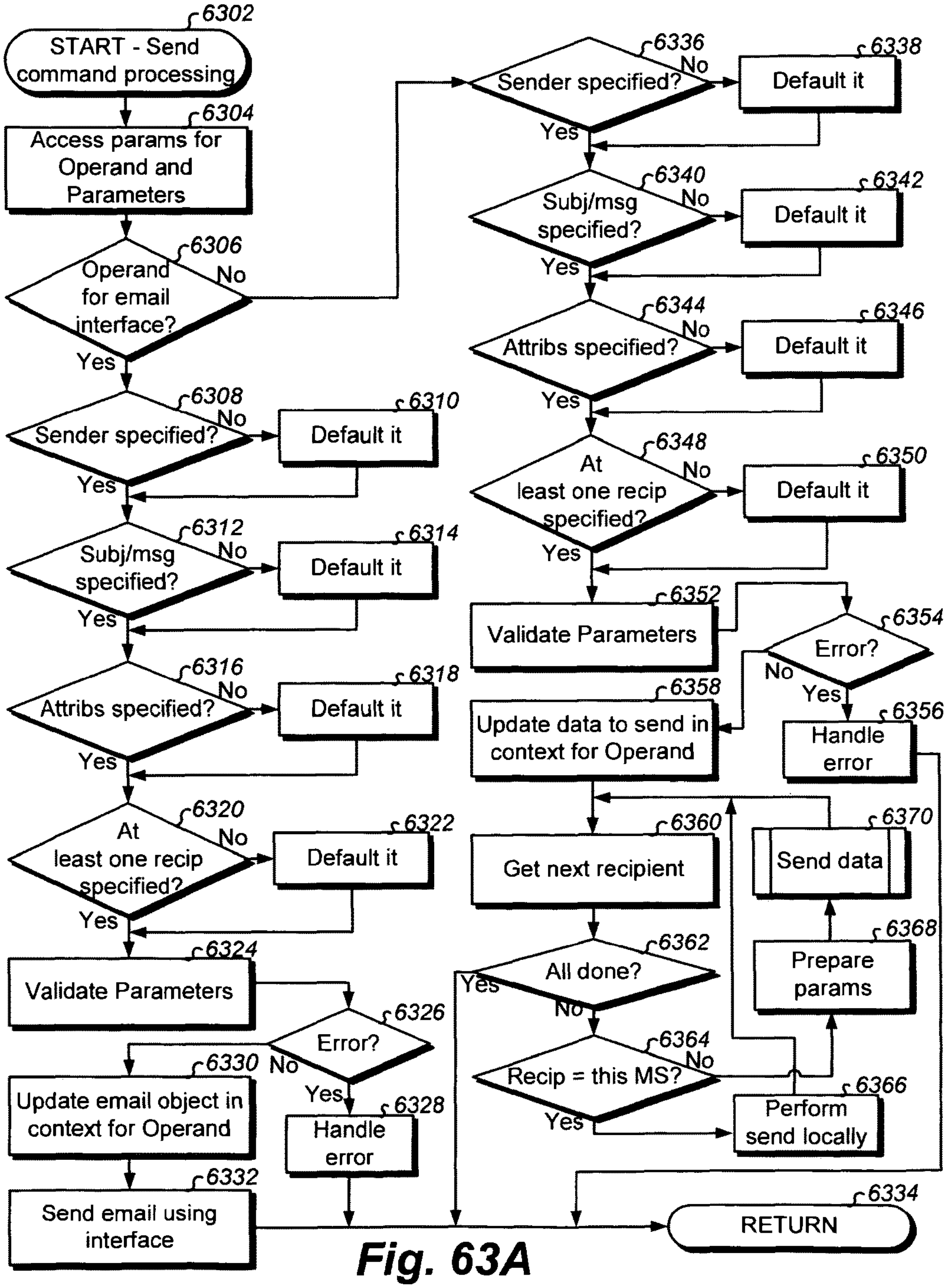

6302
START - Send command processing
6304
Access params for Operand and Parameters
6306
Operand for email interface?
No
Yes
6308
Sender specified?
No
6310
Default it
Yes
6312
Subj/msg specified?
No
6314
Default it
Yes
6316
Attribs specified?
No
6318
Default it
Yes
6320
At least one recip specified?
No
6322
Default it
Yes
6324
Validate Parameters
6326
Error?
No
Yes
6328
Handle error
6330
Update email object in context for Operand
6332
Send email using interface
6336
Sender specified?
No
6338
Default it
Yes
6340
Subj/msg specified?
No
6342
Default it
Yes
6344
Attribs specified?
No
6346
Default it
Yes
6348
At least one recip specified?
No
6350
Default it
Yes
6352
Validate Parameters
6354
Error?
No
Yes
6356
Handle error
6358
Update data to send in context for Operand
6360
Get next recipient
6362
All done?
Yes
No
6364
Recip = this MS?
No
Yes
6366
Perform send locally
6368
Prepare params
6370
Send data
6334
RETURN

Fig. 63A

Operand ↓	**PM**	**Preferred embodiment Send processing**
201	**O**	Sending an auto-dial # updates appropriate recipient MS storage so that a recipient user can subsequently auto-dial the auto-dial # with a minimal user interface action. Preferably, the recipient MS user is appropriately and immediately notified of the receipt. Preferably, the send command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent user browse of the accompanying message and a date/time stamp of when sent, and for automated speed dialing of the # in response to a user action to auto-dial. Various embodiments will save to LBX History how many times, and when, the auto-dial # was used to perform automated speed dialing.
203	**O**	Sending a web link updates appropriate recipient MS storage so a recipient user can subsequently invoke (transpose to) the link, for example in a browser, with a minimal user interface action. Preferably, the recipient MS user is appropriately and immediately notified of the receipt. Preferably, the send command data is maintained to LBX History, a historical call log (e.g. incoming), browser history data, browser favorites, or other useful storage for subsequent user browse of the accompanying message and a date/time stamp of when sent, and for invocation of the link within a MS browser in response to a user action to use the link. Various embodiments will save to LBX History how many times, and when, the weblink was invoked.
205	**E**	Sending an email causes interface to the email delivery system (e.g. SMTP API) for sending the email body parameter. In one embodiment, the body is assumed to be the body of the email. In another embodiment, the body is attached with or without attachment(s). Attachments are preferably referenced with an appropriate syntax in the body specified. In another embodiment, the body is parsed for determining and using the best delivery options. The email will arrive to a recipient like other emails. Attributes can be set as is customary for email attributes (e.g. confirmation of delivery status, special handling, NLS considerations, etc).
207	**E**	Sending an SMS message causes interface to the sms message delivery system (e.g. SMTP API) for sending the sms message. The email interface can be used provided the sms message length maximum is observed. In one embodiment, the message parameter is identical to the msg/subj parameter. In another embodiment, the two parameters are concatenated, or formed in a complimentary manner, to highlight the subj/msg parameter from the sms message. In another embodiment, only a null subj/msg is supported. The message will arrive to a recipient like other sms messages. Various attributes can be set (e.g. confirmation of delivery status, special handling, NLS considerations, etc).

Fig. 63B-1

Operand ↓	PM	Preferred embodiment Send processing
209	E	Sending a broadcast email causes interface to the email delivery system (e.g. SMTP API) for sending the email body parameter. In one embodiment, the body is assumed to be the body of the email. In another embodiment, the body is attached with or without attachment(s). Attachments are preferably referenced with an appropriate syntax in the body specified. In another embodiment, the body is parsed for determining and using the best delivery options. The email will arrive to a recipient like other emails. Various attributes can be set (e.g. special handling, NLS considerations, etc), but preferably, no confirmation of delivery is set since this is a broadcast.
211	E	Sending an SMS broadcast message causes interface to the sms message delivery system (e.g. SMTP API) for sending the sms message parameter. The email interface can be used provided the sms message length maximum is observed. In one embodiment, the message parameter is identical to the msg/subj parameter. In another embodiment, the two parameters are concatenated, or formed in a complimentary manner, to highlight the subj/msg parameter from the sms message. In another embodiment, only a null subj/msg is supported. The message will arrive to a recipient like other messages. Various attributes can be set (e.g. special handling, NLS considerations, etc), but preferably, no confirmation of delivery status is set since this is a broadcast.
213	O	Sending an indicator updates appropriate recipient MS storage so that the currently focused user interface object (e.g. window titlebar) of the MS user interface is modified with the indicator. If there are no active user interface objects in the current MS user interface, then an appropriate alert area of the currently focused interface is to display the indicator. The user can clear (remove) the indicator when desired. Preferably, the indicator is used for modifying other focused objects (e.g. titlebars) or other focused areas in the user interface so as to not get overlooked. For example, as the user navigates and surfaces/focuses new user interface objects, the indicator remains visible on the newly focused object. Preferably, the indicator is selectable by the user of the MS for showing all other send command parameters associated, as well as a date/time stamp of when sent. In other embodiments, the most recently displayed indicator is displayed in the appropriate focused area, but the user can conveniently select any indicators which were sent in history at some point in time for sought indicator information by selecting the currently displayed indicator and then requesting to browse/scroll history of previously delivered indicators (with options to see details). Preferably, the send command data is maintained to LBX History, a historical log (web page load history), or other useful storage for subsequent use. Some title bar management methods include various IBM Technical Disclosure Bulletins from 1991 through 1995 (e.g. DA8-92-0910 "Originator Identified Direct Access Mail Basket Title Bar Mechanism", DA8-93-0061 "Roving Title Bar", DA8-93-0223 "Roving Title Bar Status", etc).

Fig. 63B-2

Operand ↓	PM	Preferred embodiment Send processing
215	O	Sending an application causes invocation of the application at the recipient MS. The app parameter is preferably a fully qualified path name to the executable to start. In another embodiment, the app parameter is indirect: a path name to a "shortcut" (like a MS Windows shortcut). In another embodiment, the app parameter is an identifier string for the underlying operating system to know which application to start. The attributes parameter can be used for how to start the application, for example to flag whether to start an additional instance if the application is already running at the MS (provided multiple instances are supported). The msg/subject parameter may be useful for maintaining to LBX history useful information, along with a date/time stamp when sent, with record of the application invocation reference. An error is logged if the app parameter is not found for launch. Preferably, the invoke command data is maintained to LBX History, a historical log (web page load history), or other useful storage for subsequent use.
217	E	Sending a document causes interface to the email delivery system (e.g. SMTP API) for appropriately sending the document. The doc parameter is preferably a fully qualified path name, or suitable reference, to the document which may have a document type (e.g. by file extension, document parse, or document location). The document type is used for setting proper email attachment settings and perhaps the attributes parameter. Depending on the document type, the document may form the email body or be an attachment. The email will arrive to a recipient like other emails. Various attributes can be set (e.g. confirmation of delivery status, special handling, NLS considerations, etc).
219	E	Sending a file causes interface to the email delivery system (e.g. SMTP API) for appropriately sending the file. The path parameter is preferably a fully qualified path name, or suitable reference, to the file which should have a file type (e.g. by file extension, file parse, or file location). The file type is used for setting proper email attachment settings and perhaps the attributes parameter. Depending on the file type, the file may form the email body or be an attachment. The email will arrive to a recipient like other emails. Various attributes can be set (e.g. confirmation of delivery status, special handling, NLS considerations, etc).
221	O	Sending content causes the content to be sent to the recipient MS in a manner which is appropriate for where the content is stored and how it is to be subsequently presented. The content parameter is one that cannot be classified in the other operands, but is content for presentation nevertheless. Examples include special data records (e.g. extern variable name), content data memory locations (e.g. programmatic variable), or files containing a customizably processed format. Methods of displaying the content include audio and/or visual using applicable MS capabilities. Preferably, the send command data is maintained to LBX History, a historical content log, or other useful storage for subsequent user browse of the accompanying content and a date/time stamp of when sent, and for presentation of the content in response to an applicable user action. Attributes may be set for special content handling.

Fig. 63B-3

Operand ↓	PM	Preferred embodiment Send processing
223	O	Sending a Database (DB) object causes the DB object to be sent to the recipient MS in a manner which is appropriate for subsequent import DB or table(s). The DB-obj parameter takes on many syntaxes for sending any subset of a database object, such as an entire database, table(s), certain rows, certain columns, etc. In one embodiment, a qualified database form is used such as: Owner:DatabaseName:TableName for sending the entire table (can use table name wildcard for multiple tables). In another embodiment, Owner:DatabaseName:"...SQL query... shall return the data that is to be sent, preferably in a comma delimited or tab delimited form (as specified in the attributes parameter). Preferably, the send command data is maintained to LBX History, a database log, or other useful storage (subj/msg to document the transaction) for subsequent user browse of the accompanying data and a date/time stamp of when sent, and for DB query manager browse of the data in response to a user action for browse. One preferred embodiment enables the data for easy import to a variety of database destinations, preferably via the same DB query mgr interface(s) used for browsing.
225	O	Sending data causes reading the current value of the data at the MS where the send command action is being executed and then sending the current value to the recipient MS for informative purposes. In the preferred embodiment, the data is a global system variable visible to all processes of a MS operating system. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). An AppTerm uses record 5300 for access. Depending on the embodiment, data may be that which is contained in a program data segment, stack segment, and/or extra segment. There can be unique syntaxes for specifying which type of data is being sought (e.g. "S:dataname"). In the preferred embodiment, all occurrences found on the MS and information including its value about the occurrence is presented to the user. In one embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. Preferably, the data value is maintained to LBX History, a historical log, or other useful storage for subsequent user browse, or programmatic access, of the data variable name, its value and date/time stamp of when sent, and for presentation of the data value in response to a user action to show it.
227	O	Sending a semaphore causes reading the current value of the semaphore at the MS where the send command action is being executed and then sending the current value to the recipient MS for informative purposes. In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing (e.g. RAM semaphore). Preferably, the semaphore value (cleared or set) is maintained to LBX History, a historical log, or other useful storage for subsequent user browse, or programmatic access, of the semaphore name, its value and date/time stamp of when sent, and for presentation of the semaphore value in response to a user action to show it.

Fig. 63B-4

Operand ↓	PM	Preferred embodiment Send processing
229	E	Sending a directory causes interface to the email delivery system (e.g. SMTP API) for sending the directory. In one embodiment, the directory is assumed to be the body of the email (e.g. when attributes parameter indicates it is to be a description only of the directory) for sending information about the directory such as # files, nesting of folders, sizes, and any useful file system characteristic(s) or statistics of the directory. In another embodiment, or as specified with an additional parameter (or in attributes), the directory is compressed and encoded as an attachment. In another embodiment, the directory is sent as individually attached files (as indicated to send that way by new or attributes parameter). The email will arrive to a recipient like other emails. The attribute parameter can be used for conventional email attributes as well as new attributes which affect directory data processing.
231	O	Sending an application context causes invocation of the application at the recipient MS and then executing a macro within the application context. The app parameter is preferably a fully qualified path name to the executable to start. In another embodiment, the app parameter is indirect: a path name to a "shortcut" (like a MS Windows shortcut). In another embodiment, the app parameter is an identifier string for the underlying operating system to know which application to start. The macro parameter is preferably a file, path, or accessible variable name containing a set of keystrokes that can be directed to standard/user-interface input. In another embodiment, the macro parameter is a prerecorded user input scenario (for play after application launched -- pulldown selections, mouse droppings, clicks, etc) captured to a file or stored in an accessible variable name. The attributes parameter can be used for how to start the application, for example to flag whether to start an additional instance if the application is already running at the MS (provided multiple instances are supported), and to specify the type of macro parameter being specified, or to specify a speed for processing individuals of the macro. The msg/subject parameter may be useful for maintaining to LBX history useful information with record of the application context invocation reference. An error is logged if the app parameter is not found for launch.
233	E	Sending the focused user interface object causes interface to the email delivery system (e.g. SMTP API) for sending an image (preferably .JPG) of the currently focused user interface object as an attachment. The "<alt><prtscrn>" constant string parameter is a syntactical string representation for the keystroke sequence for performing the MS focused user interface capture action. A similar syntax can be used to specify a different keystroke sequence (1st parameter) for the same functionality. The email will arrive to a recipient like other emails. The attributes parameter can be set for which format to send, in which case a conversion may take place prior to sending (depends on embodiment). Various attributes can be set (e.g. confirmation of delivery status, special handling, NLS considerations, etc).

Fig. 63B-5

Operand ↓	PM	Preferred embodiment Send processing
235	O	Sending user interface control causes redirecting the keystroke macro to input of the recipient MS as if it were entered by the MS user. The macro parameter is preferably a file, path, or accessible variable name containing a set of keystrokes that can be directed to standard user-interface/input. In another embodiment, the macro is a prerecorded user input scenario (for play after application launched -- pulldown selections, mouse droppings, clicks, etc) captured to a file or stored in an accessible variable name. The attributes parameter can be used for whether or not to first display the subj/msg to the recipient MS user for user acknowledgement, or cancellation, prior to executing the macro at the MS. This allows the user time to get the MS user interface in a desirable state if necessary for running the macro, and to see information of the origination (i.e. Parameters). The msg/subject parameter may be useful for maintaining to LBX history information with a record of user interface control sent.
237	O	Sending input causes redirecting the input to the iodev parameter input device stream of the recipient MS as if it were entered by the MS user, or programmatically specified to the iodev I/O device parameter by a data processing system process. The input parameter is preferably a file, path, or accessible variable name containing a datastream (e.g. macro) recognizable by the iodev connected device. The attributes parameter can be used for whether or not to first display the subj/msg to the recipient MS user for user acknowledgement, or cancellation, prior to redirecting the input parameter datastream at the MS, or to specify a speed for processing individuals of the input. This allows the user time to get the MS user interface, and any iodev devices, in a desirable state if necessary for running the input, and to see information of the origination (i.e. Parameters). The msg/subject parameter may be useful for maintaining to LBX history information with a record of the user interface control having been sent.
239	O	Sending output causes redirecting the output to the iodev parameter output device stream of the recipient MS as if it were entered by the MS user, or programmatically specified to the iodev I/O device parameter by a data processing system process. The output parameter is preferably a file, path, or accessible variable name containing a datastream (e.g. macro) recognizable by the iodev connected device. The attributes parameter can be used for whether or not to first display the subj/msg to the recipient MS user for user acknowledgement prior to redirecting the output parameter datastream at the MS, or to specify a speed for processing individuals of the output. This allows the user time to get the MS user interface, and any iodev devices, in a desirable state if necessary for running the output, and to see information of the origination (i.e. Parameters). The msg/subject parameter may be useful for maintaining to LBX history information with a record of the user interface control having been sent.

Fig. 63B-6

Operand ↓	PM	Preferred embodiment Send processing
241	O	Sending an alert updates appropriate recipient MS storage so that a recipient MS alerter process can pick up the alert and then alert the user. There are a variety of alert processes, the most basic of which monitors incoming messages and posts them to the user in an alerting manner. In one embodiment, the alert parameter is identical to the msg/subj parameter. In another embodiment, the two parameters are concatenated, or formed in a complimentary manner, to highlight the subj/msg parameter from the alert message. In another embodiment, only a null subj/msg is supported. The attributes parameter can be for special treatment of the alert by an alerter process.
243	O	See Notify Command for identical processing.
245	O	See Notify Command for identical processing.
247	O	See Notify Command for identical processing.
249	O	See Notify Command for identical processing.
251	O	Sending a calendar object causes interface to the recipient MS calendar/scheduling system for sending/scheduling the calendar object parameter. The calobj parameter contains the date/time stamp of when to schedule the object, or a special syntax constant for "now", "first available per recipient and sender availability", "by end of the week pending availability", or other reasonable constants for when to schedule the calendar object. In one embodiment, the calendar object is assumed to be a newly scheduled calendar item for placement to the calendar of recipients. In another embodiment, the calendar object (e.g. data or file containing parsable syntax) contains directives for what actions exactly to perform to the calendar application interface. In another embodiment, the email system is the transport to deliver the calendar object or calendar actions to recipients. Attributes can be set as is customary for calendar entries (attendance required, emergency meeting, recurring/weekly/monthly meeting, etc). The attributes parameter may be used for performing other actions/functions in the calendaring interface.
253	O	Sending an address book (AB) object causes interface to the AB system for sending/entering the AB object parameter at the recipient MS. In one embodiment, the AB object is assumed to be a newly entered AB entry (e.g. contact reference name) for creation in the AB of recipients. In another embodiment, the AB object parameter contains directives (e.g. data or file containing parsable syntax) for what actions exactly to perform to the AB application interface. Attributes can be set as may be customary for AB entries (customer, peer, manager, friend, family, etc). The attributes parameter may be used for performing other actions/functions in the AB interface.
. . .		

Fig. 63B-7

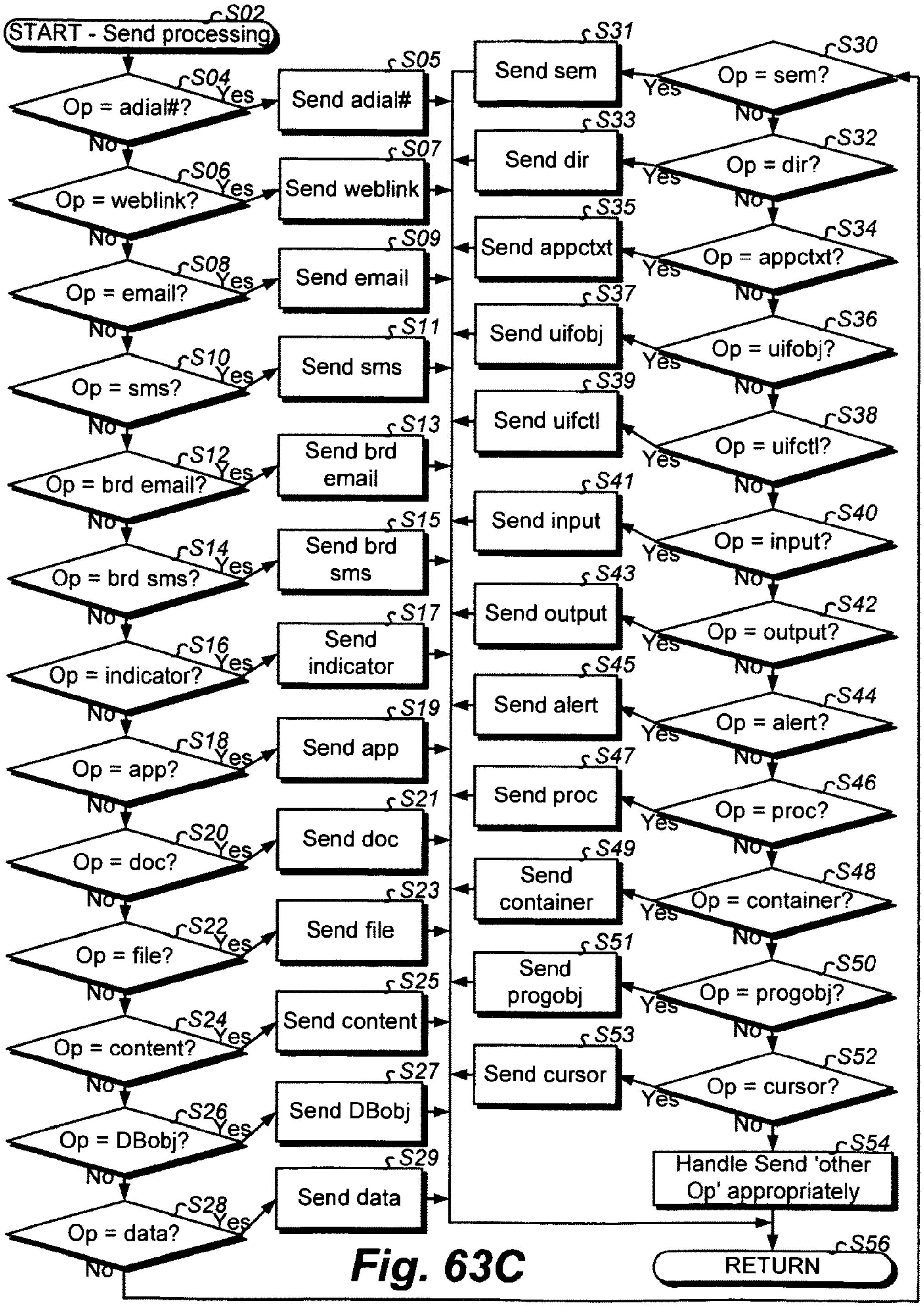

START - Send processing
S02
Op = adial#?
S04
Yes
No
Send adial#
S05
Op = weblink?
S06
Send weblink
S07
Op = email?
S08
Send email
S09
Op = sms?
S10
Send sms
S11
Op = brd email?
S12
Send brd email
S13
Op = brd sms?
S14
Send brd sms
S15
Op = indicator?
S16
Send indicator
S17
Op = app?
S18
Send app
S19
Op = doc?
S20
Send doc
S21
Op = file?
S22
Send file
S23
Op = content?
S24
Send content
S25
Op = DBobj?
S26
Send DBobj
S27
Op = data?
S28
Send data
S29
Op = sem?
S30
Send sem
S31
Op = dir?
S32
Send dir
S33
Op = appctxt?
S34
Send appctxt
S35
Op = uifobj?
S36
Send uifobj
S37
Op = uifctl?
S38
Send uifctl
S39
Op = input?
S40
Send input
S41
Op = output?
S42
Send output
S43
Op = alert?
S44
Send alert
S45
Op = proc?
S46
Send proc
S47
Op = container?
S48
Send container
S49
Op = progobj?
S50
Send progobj
S51
Op = cursor?
S52
Send cursor
S53
Handle Send 'other Op' appropriately
S54
RETURN
S56

Fig. 63C

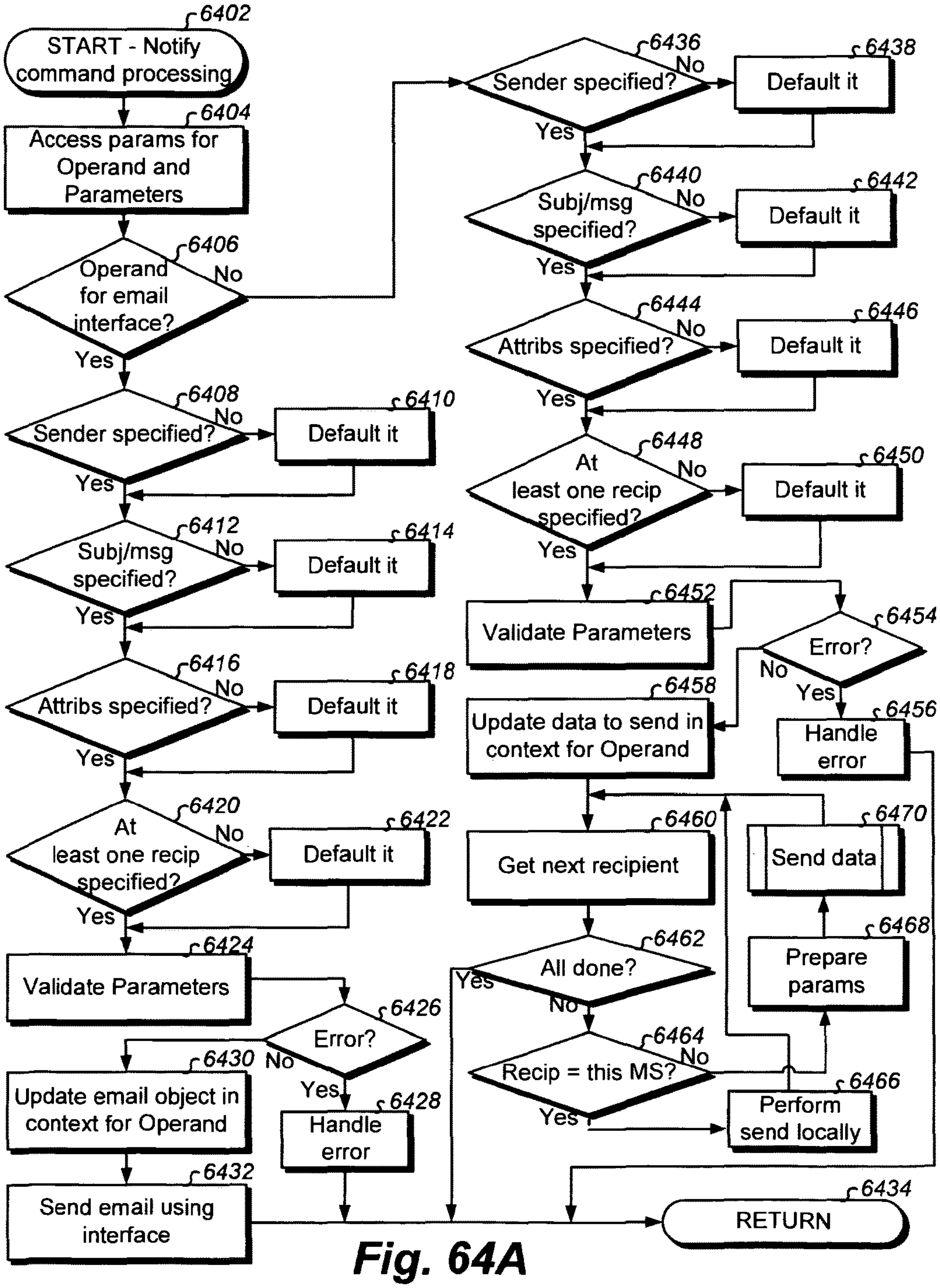

6402
START - Notify command processing
6404
Access params for Operand and Parameters
6406
Operand for email interface?
No
Yes
6408
Sender specified?
No
Yes
6410
Default it
6412
Subj/msg specified?
No
Yes
6414
Default it
6416
Attribs specified?
No
Yes
6418
Default it
6420
At least one recip specified?
No
Yes
6422
Default it
6424
Validate Parameters
6426
Error?
No
Yes
6428
Handle error
6430
Update email object in context for Operand
6432
Send email using interface
6436
Sender specified?
No
Yes
6438
Default it
6440
Subj/msg specified?
No
Yes
6442
Default it
6444
Attribs specified?
No
Yes
6446
Default it
6448
At least one recip specified?
No
Yes
6450
Default it
6452
Validate Parameters
6454
Error?
No
Yes
6456
Handle error
6458
Update data to send in context for Operand
6460
Get next recipient
6462
All done?
Yes
No
6464
Recip = this MS?
No
Yes
6466
Perform send locally
6468
Prepare params
6470
Send data
6434
RETURN

Fig. 64A

Operand ↓	PM	Preferred embodiment Notify processing
201	O	Notifying with an auto-dial # automatically performs call processing to auto-dial the auto-dial #. Preferably, the recipient MS user is called by the MS as a normal phone call would be made. In one embodiment, multiple recipients are called "back to back" after the previous recipient call terminates. In another embodiment, a multiple line party call is made with an automated manner with all recipients. The attributes parameter can indicate which embodiment to use, and can be used for specialized call processing (collect, prepaid account check, hide caller id, etc). Preferably, the notify command data and call data is maintained to LBX History, a historical call log (e.g. incoming), with the accompanying subj/msg and a date/time stamp of when sent, and for future repeated automated speed dialing of the # in response to a user action to auto-dial. Various embodiments will save to LBX History how many times, and when, the auto-dial # was used to perform automated speed dialing, along with call details such as direction of call, parties to the call, features of the call, or other call characteristics. In one embodiment, the recipient ID is one to one with the called #. In another embodiment, the recipient ID is used to find the associated called number. Preferably, an existing API is used to accomplish processing. Automatic dialing through a variety of interfaces is well known in the art, and depends on the software development environment. Conventional processing side affects of automated calling should occur like the action was manual (e.g. log update).
203	O	Notifying with a web link automatically invokes (transposes to) at the recipient MS the link, for example in a browser, with a minimal (if any) user interface action. In one embodiment, the link includes URL parameter(s) (e.g. ?p1=xyz). In another embodiment, all recipients are passed to the link with appended URL parameter(s) (e.g. ?ids=Recip1;Recip2;...RecipN). An alternate embodiment fires form variables to the loaded page with the same URL variables. The attributes parameter can indicate which embodiment to use, and how to invoke the link (e.g. use currently focused window, use an active browser window, spawn new browser window, etc). Preferably, the notify command data is maintained to LBX History, a historical call log (e.g. incoming), browser history data, browser favorites, or other useful storage for subsequent user browse of the accompanying subj/msg and a date/time stamp of when sent, and for invocation of the link within a MS browser in response to a user action to use the link again in the future. Various embodiments will save to LBX History how many times, and when, the weblink was invoked.
205	E	See Send Command for identical processing.
207	E	See Send Command for identical processing.
209	E	See Send Command for identical processing.
211	E	See Send Command for identical processing.
213	O	See Send Command for identical processing.
215	O	See Send Command for identical processing.

Fig. 64B-1

Operand ↓	PM	Preferred embodiment Notify processing
217	E	See Send Command for identical processing.
219	E	See Send Command for identical processing.
221	O	Notifying with content causes the content to be presented at the recipient MS upon delivery in a manner which is appropriate for the content type. The content parameter is one that cannot be classified in the other operands, but is content for presentation nevertheless. Examples include special data records (e.g. extern variable name), content data memory locations (e.g. programmatic variable), or files containing a customizably processed format. Methods of displaying the content include audio and/or visual using applicable MS capabilities. Preferably, the notify command data is maintained to LBX History, a historical content log (e.g. incoming), browser history data, or other useful storage for subsequent user browse of the accompanying content and a date/time stamp of when sent, and for presentation of the content. Various embodiments will save to LBX History how many times, and when, the content was presented.
223	O	Notifying with a Database (DB) object causes the DB object (i.e. qualified database with access query string) to be modified with the query parameter. The query parameter is used to perform any query against the specified DB-database (DB-obj), preferably a query that only returns a return code (e.g. causes alteration). Preferably, the notify command data is maintained to LBX History, a database log (e.g. incoming), or other useful storage for subsequent query use and a date/time stamp of when sent, and for DB query manager browse/use of the query in response to an applicable user action. Other params are for documentary purposes when information is saved. In some embodiments, an appropriate SQL client interface (e.g. SQLNET API) is used to carry out processing, or a suitable DB API is used.
225	O	Notifying data causes modifying the value of the data at the recipient MS (set data to value). An error can result if the data is not resolvable for the attempt. In the preferred embodiment, the data is a global system variable visible to all processes of a MS operating system. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). An AppTerm uses record 5300 for access. Preferably, the data affected is maintained to LBX History, a historical log (e.g. incoming), or other useful storage for subsequent user browse, or programmatic access, of the data variable name, its before and after values and date/time stamp of when sent, and for presentation of the data value in response to a user action to show it.
227	O	Notifying a semaphore causes modifying the value of the semaphore at the recipient MS. In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing (e.g. RAM semaphore). Preferably, the semaphore value before and after setting is maintained to LBX History, a historical log (e.g. incoming), or other useful storage for subsequent user browse, or programmatic access, and for presentation of the semaphore information in response to a user action to show it.

Fig. 64B-2

Operand ↓	PM	Preferred embodiment Notify processing
229	E	See Send Command for identical processing.
231	O	See Send Command for identical processing.
233	E	See Send Command for identical processing.
235	O	See Send Command for identical processing.
237	O	See Send Command for identical processing.
239	O	See Send Command for identical processing.
241	O	Notifying with an alert presents the alert to each recipient MS. Depending on attributes parameter settings, the alert may be asynchronously presented to an alert area, synchronously alerted and requiring a user action to acknowledge, logged to special file, or other reasonable alert method(s). Fig. 75B processing will cause the alert to be presented to the MS user. In one embodiment, the alert parameter is identical to the msg/subj parameter. In another embodiment, the two parameters are concatenated, or formed in a complimentary manner, to highlight the subj/msg parameter from the alert message. In another embodiment, only a null subj/msg is supported. Various embodiments will support different alert content types and applicable processing as indicated by the attributes parameter. Preferably, an appropriate API is made available for processing.
243	O	Notifying a process causes sending an operating system signal (see UNIX signaling) to the process with Process ID (PID) of the pid parameter. A numeric value parameter (e.g. 0 or 1) may be communicated with the signal. Depending on attributes parameter settings, another embodiment accesses the pid parameter as a process identifier parameter which is used to lookup the operating system PID prior to signaling. The msg/subject parameter may be useful for maintaining to LBX history useful information, along with a date/time stamp when sent, with record of the application invocation reference. An error can be logged if the process is not found for signaling.
245	O	Notifying a container causes launch of a MS file manager to examine the contents of a MS system container having the path in the container parameter (e.g. c:\dir1\subdir3). The attributes parameter can be used for how to start the file manager, for example to flag whether to start an additional instance if the file manager is already running at the MS (provided multiple instances are supported), otherwise an existing instance is updated for the container, or a new instance is started for the container. The msg/subject parameter may be useful for maintaining to LBX history useful information, along with a sent date/time stamp, with record of the application invocation reference. An error can be logged if the file manager is not found for launch, or if the container is invalid.

Fig. 64B-3

Operand ↓	**PM**	**Preferred embodiment Notify processing**
247	**O**	Notifying a program object causes acting on a specified program object (per attribute parameter) with the specified data at the recipient MS. The progobj parameter is the linked run time symbolic name accessible to charter processing of the present disclosure for third party plug-in processing. The progobj parameter can be a variable name, function name, object name, queue name, procedure name, or semaphore name accessed at run time by the symbolic name evaluation during charter processing. The binary data parameter is used to modify the program object (variable name set Least Significant Bit (LSB) to Most significant Bit (MSB) right to left intuitive Motorola processing byte/bit order until bits set or unmatched, function name invoked with respective data bytes pushed to the stack prior to invocation, object name data public data area initialized with the data parameter on a byte to byte basis, queue name entry inserted using the data parameter as a typecast data record of bits, procedure name invoked with respective data bytes pushed to the stack prior to invocation, or semaphore name set with clear for a null data parameter, else a set action). Alternately, an Intel reverse byte order can be used to apply the data Parameter. The attributes parameter indicates which variety of progobj is specified, and can be used to indicate a byte order data mapping method to use. The msg/subject parameter may be useful for maintaining to LBX history useful information, along with a date/time stamp when sent, with record of the program object invocation. An error can be logged if the progobj parameter is not resolvable. Appropriate MS O/S interfaces are used.
249	**O**	Notifying a cursor causes modifying a recipient MS user interface cursor in accordance with direction by the attributes parameter. The cursor parameter can be a suitable cursor bitmap file reference, suitable animated cursor file, predefined appearance type, or predefined behaving cursor. The attributes parameter further distinguishes which cursor modification is being requested. The msg/subject parameter may be useful for maintaining to LBX history useful information, along with a date/time stamp when sent. An error is logged if there is no active cursor in the user interface. An appropriate MS API is used, depending on the development environment.
251	**O**	See Send Command for identical processing.
253	**O**	See Send Command for identical processing.
. . .		

Fig. 64B-4

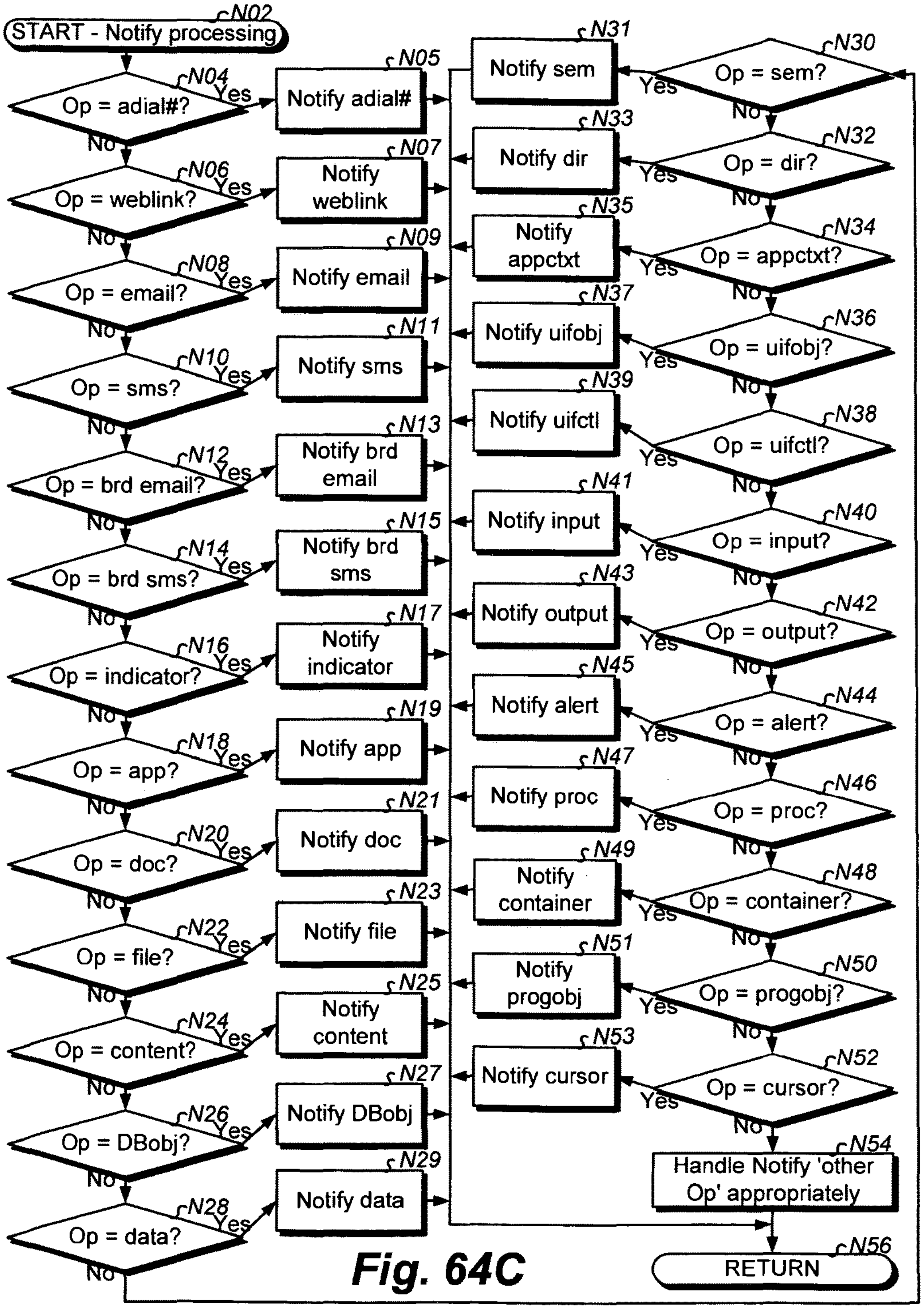

N02
START - Notify processing
N04
Op = adial#?
Yes
No
N05
Notify adial#
N06
Op = weblink?
N07
Notify weblink
N08
Op = email?
N09
Notify email
N10
Op = sms?
N11
Notify sms
N12
Op = brd email?
N13
Notify brd email
N14
Op = brd sms?
N15
Notify brd sms
N16
Op = indicator?
N17
Notify indicator
N18
Op = app?
N19
Notify app
N20
Op = doc?
N21
Notify doc
N22
Op = file?
N23
Notify file
N24
Op = content?
N25
Notify content
N26
Op = DBobj?
N27
Notify DBobj
N28
Op = data?
N29
Notify data
N30
Op = sem?
N31
Notify sem
N32
Op = dir?
N33
Notify dir
N34
Op = appctxt?
N35
Notify appctxt
N36
Op = uifobj?
N37
Notify uifobj
N38
Op = uifctl?
N39
Notify uifctl
N40
Op = input?
N41
Notify input
N42
Op = output?
N43
Notify output
N44
Op = alert?
N45
Notify alert
N46
Op = proc?
N47
Notify proc
N48
Op = container?
N49
Notify container
N50
Op = progobj?
N51
Notify progobj
N52
Op = cursor?
N53
Notify cursor
N54
Handle Notify 'other Op' appropriately
N56
RETURN

Fig. 64C

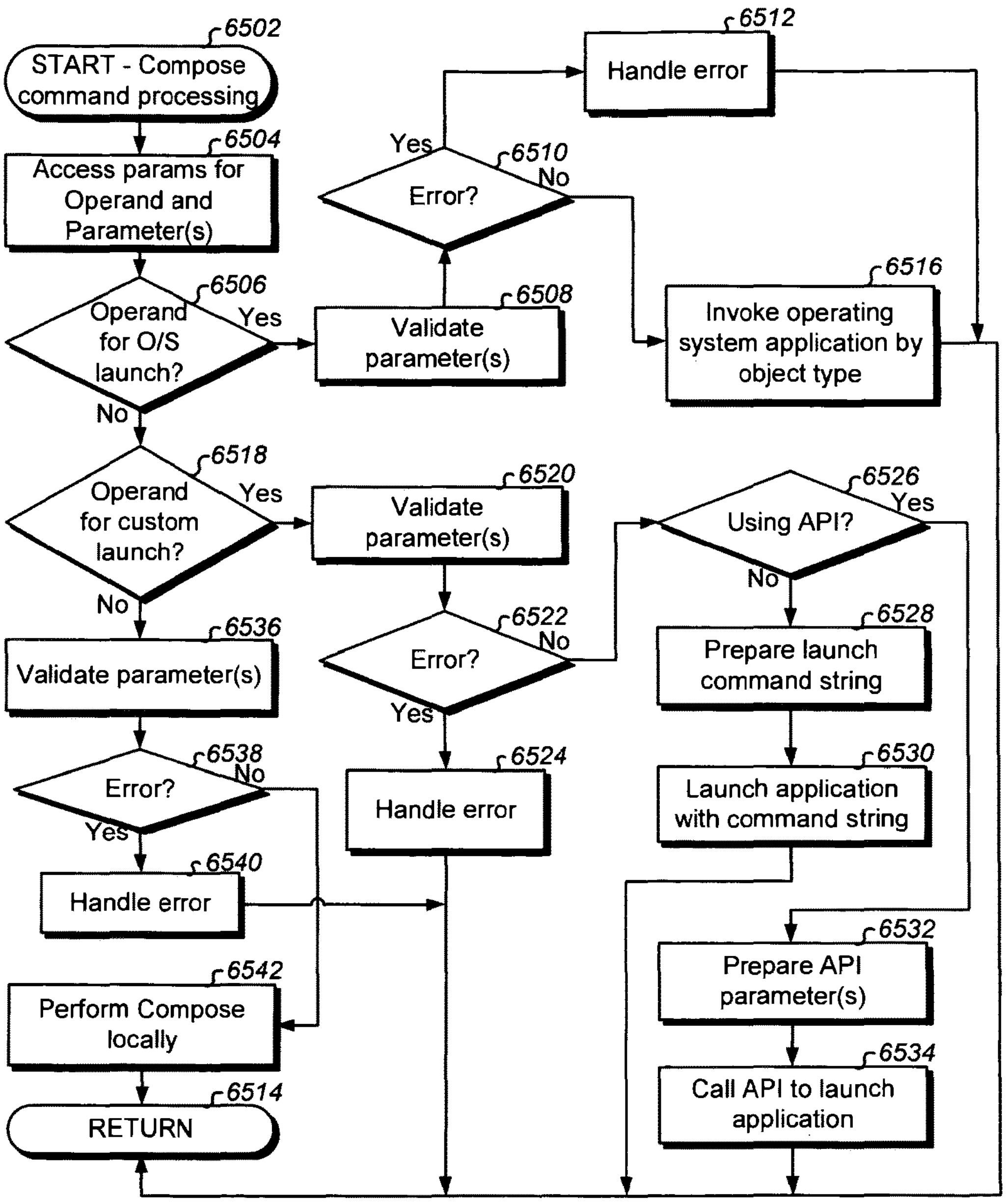

6502
START - Compose command processing
6504
Access params for Operand and Parameter(s)
6506
Operand for O/S launch?
Yes
No
6508
Validate parameter(s)
6510
Error?
Yes
No
6512
Handle error
6516
Invoke operating system application by object type
6518
Operand for custom launch?
Yes
No
6520
Validate parameter(s)
6522
Error?
No
Yes
6524
Handle error
6526
Using API?
Yes
No
6528
Prepare launch command string
6530
Launch application with command string
6532
Prepare API parameter(s)
6534
Call API to launch application
6536
Validate parameter(s)
6538
Error?
No
Yes
6540
Handle error
6542
Perform Compose locally
6514
RETURN

Fig. 65A

Operand ↓	**PM**	**Preferred embodiment Compose processing**
201	**C**	Composing an auto-dial # launches the MS phone number calling interface with the auto-dial # parameter defaulted for making the call. Once launched, the user can make a very simple confirmation action for placing the call to the auto-dial #. Call processing takes place as though the user manually launched the dialing application, entered the auto-dial # and then is ready to decide if the call should be placed. Appropriate MS storage is updated and subsequently processed as though the user had entered the # for calling manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. outgoing when call placed), or other useful storage for subsequent use.
203	**S**	Composing a web link launches a MS browser and defaults the link as though the user had entered it manually. The user can subsequently invoke (transpose to) the link if desired with a minimal action (e.g. click ok). The link may include appended URL parameters (e.g. "?v=yes&T=go" for customized web page processing). An alternate embodiment can fire form variables for active web page processing using URL parameters specified or using the params parameter. Processing takes place as though the user manually launched the browser application, entered the weblink and then is ready to decide if the link should be invoked. Appropriate MS storage is updated and subsequently processed as though the user had entered the weblink in the browser manually. Preferably, the compose command data is maintained to LBX History, a historical log (web page load history when invoked), or other useful storage for subsequent use.
205	**C**	Composing an email causes interface to the email delivery system for invoking the create email interface and defaulting the appropriate email fields with the passed parameters. The user can subsequently send the email with little effort, or after optional modification, with a minimal action (e.g. click ok). Processing takes place as though the user manually launched the create email application, entered the fields of the email form with passed parameters, and then is ready to decide if the email should be sent, or further edited, or possibly cancelled. Appropriate MS storage is updated and subsequently processed as though the user manually started the create email interface and entered the email information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use. A standard email POP or mailbox interface is preferably used. The email will arrive to a recipient like other emails. Various attributes can be set (e.g. confirmation of delivery status, special handling, NLS considerations, etc).

Fig. 65B-1

Operand ↓	PM	Preferred embodiment Compose processing
207	C	Composing an sms message causes interface to an appropriate messaging delivery system for invoking the create message interface and defaulting the appropriate message fields with the passed parameters. The user can subsequently send the message with little effort, or after optional modification, with a minimal action (e.g. click ok). Processing takes place as though the user manually launched the create message application, entered the fields of the messaging form with passed parameters, and then is ready to decide if the message should be sent, or further edited, or possibly cancelled. Appropriate MS storage is updated and subsequently processed as though the user manually started the create message interface and entered message information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use. A standard email POP or mailbox interface, or a similar messaging interface, can be used. The message will arrive to a recipient like other sms messages. Various sms message attributes may be set (e.g. confirmation of delivery status, special handling, NLS considerations, etc).
209	C	Composing a broadcast email causes interface to the email delivery system for invoking the create email interface and defaulting the appropriate email fields with the passed parameters. The user can subsequently send the email with little effort, or after optional modification, with a minimal action (e.g. click ok). Processing takes place as though the user manually launched the create email application, entered the fields of the email form with passed parameters, and then is ready to decide if the email should be sent, or further edited, or possibly cancelled. Appropriate MS storage is updated and subsequently processed as though the user manually started the create email interface and entered the email information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use. A standard email POP or mailbox interface is preferably used. The email will arrive to a recipient like other emails. Various attributes can be set (e.g. special handling, NLS considerations, etc), but preferably, no confirmation of delivery status is requested in attributes since this is a broadcast.

Fig. 65B-2

Operand ↓	PM	Preferred embodiment Compose processing
211	C	Composing a broadcast sms message causes interface to an appropriate messaging delivery system for invoking the create message interface and defaulting the appropriate message fields with the passed parameters. The user can subsequently send the message with little effort, or after optional modification, with a minimal action (e.g. click ok). Processing takes place as though the user manually launched the create message application, entered the fields of the messaging form with passed parameters, and then is ready to decide if the message should be sent, or further edited, or possibly cancelled. Appropriate MS storage is updated and subsequently processed as though the user manually started the create message interface and entered message information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use. A standard email POP or mailbox interface, or a similar messaging interface, can be used. The message will arrive to a recipient like other messages. Various attributes can be set (e.g. special handling, NLS considerations, etc), but preferably, no confirmation of delivery status is requested in attributes since this is a broadcast.
213	O	See Send Command for identical processing.
215	C	Composing an application causes invocation of the application at the MS. The app parameter is preferably a fully qualified path name to the executable to start. In another embodiment, the app parameter is indirect: a path name to a "shortcut" (like a MS Windows shortcut). In another embodiment, the app parameter is an identifier string for the underlying operating system to know which application to start. The params parameter can be used for command line, or string to append, or pass, to the app/path parameter, for how to start the application (e.g. with parameters). Processing takes place as though the user manually launched the application (and with any optional params). Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
217	S	Composing a document causes invocation of the appropriate application at the MS in accordance with the object type as though the user selected the document for automatically being associated to the correct application when opening the document for composing (e.g. edit/manage) it. The doc parameter may be preferably a fully qualified path name to the document. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
219	S	Composing a file causes invocation of the appropriate application at the MS in accordance with the file type of the fully qualified path name of the file as though the user selected the file for automatically being associated to the correct application when opening the document. Processing takes place as though the user manually launched the application for the specified file to compose it. The path parameter is preferably a fully qualified path name to the file. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.

Fig. 65B-3

Operand ↓	**PM**	**Preferred embodiment Compose processing**
221	**O**	Composing content causes invocation of the appropriate application at the MS in accordance with the content as though the user selected the content for automatically being associated to the correct application when opening the content. Processing takes place as though the user manually launched the applicable application for the content for composing (e.g. manage/edit of) it. The path parameter is preferably a fully qualified specification to the content. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
223	**C**	Composing a DB object causes invocation of the appropriate database query manager DB object creation interface to a context complementary to the type of DB object as though the user started the query manager and manually entered the DB object for creation. Processing takes place as though the user manually launched the query manager, entered the fields of the database object form, and then is ready to further work with the starting template of DB object information. In one embodiment, the DB-obj parameter contains directives for automatically populating specified data to a particular Query Manager create object interface. In another embodiment, the DB-obj parameter is specified for an existing DB object for then being opened by the query manager for further review or work. Appropriate MS storage is updated and subsequently processed as though the user had entered information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use.
225	**O**	Composing data causes modifying the value of the data at the MS (analogous to a Notify data action -- set data to value). An error can result if the data is not resolvable for the attempt. In the preferred embodiment, the data is a global system variable visible to all processes of a MS operating system. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). A recognized AppTerm causes access to record 5300 for proper semaphore synchronized access. Preferably, the data affected is maintained to LBX History, a historical log (e.g. incoming), or other useful storage for subsequent user browse, or programmatic access, of the data variable name, its before and after values and date/time stamp of when sent, and for presentation of the data value in response to a user action to show it.
227	**O**	Composing a semaphore causes modifying the value of the semaphore at the MS (analogous to a Notify sem action -- set sem to value). An error can result if the semaphore is not resolvable for the attempt. In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing (i.e. RAM semaphore). Preferably, the semaphore value before and after setting is maintained to LBX History, a historical log (e.g. incoming), or other useful storage for subsequent user browse, or programmatic access, and for presentation of the semaphore information in response to a user action to show it.

Fig. 65B-4

Operand ↓	PM	Preferred embodiment Compose processing
229	S	Composing a directory causes invocation of the appropriate application (e.g. file system manager) at the MS as though the user selected the directory for automatically being associated to the correct file management application when opening the directory. Processing takes place as though the user manually launched the application for working with the directory. The path parameter is preferably a fully qualified path name to the directory. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
231	C	Composing an application context causes invocation of the application at the MS and then executing a macro within the application context (analogous to a Send app context action). The app parameter is preferably a fully qualified path name to the executable to start. In another embodiment, the app parameter is indirect: a path name to a "shortcut" (like a MS Windows shortcut). In another embodiment, the app parameter is an identifier string for the underlying operating system to know which application to start. The macro parameter is preferably a file, path, or accessible variable name containing a set of keystrokes that can be directed to standard/user-interface input. In another embodiment, the macro parameter is a prerecorded user input scenario (for play after application launched -- pulldown selections, mouse droppings, clicks, etc) captured to a file or stored in an accessible variable name. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
233	S	Composing a focused user interface object causes invocation of the appropriate application (e.g. graphic application by file type embodiment .jpg, .gif, etc) at the MS as though the user manually captured the focused user interface object (e.g. Alt-Prtscrn) using the first command string syntax parameter, invoked the correct graphical application to open for the captured image, and is ready for save of the file, or for further editing. Processing takes place as though the user manually launched the application for the specified file. The embodiment's file type preference may influence which application is to be launched. The first parameter can be used to change the keystroke sequence for capture. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
235	O	Composing user interface control causes redirecting the keystroke macro to user interface input of the MS as if it were entered by the MS user (analogous to a Send user interface control action). The macro parameter is preferably a file, path, or accessible variable name containing a set of keystrokes that can be directed to standard input. In another embodiment, the macro is a prerecorded user input scenario (for play after application launched -- pulldown selections, mouse droppings, clicks, etc) captured to a file or stored in an accessible variable name. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.

Fig. 65B-5

Operand ↓	PM	Preferred embodiment Compose processing
237	O	Composing input causes redirecting the input to the iodev parameter input device stream of the MS as if it were entered by the MS user, or programmatically specified to the iodev I/O device parameter by a data processing system process (analogous to a Send input action). The input parameter is preferably a file or accessible variable name containing a datastream recognizable by the iodev connected device. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
239	O	Composing output causes redirecting the output to the iodev parameter output device stream of the MS as if it were entered by the MS user, or programmatically specified to the iodev I/O device parameter by a data processing system process (analogous to a Send output action). The output parameter is preferably a file or accessible variable name containing a datastream recognizable by the iodev connected device. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
241	S	Composing an alert causes invocation of the appropriate alerter application at the MS as though the user selected the alert application, manually entered the alert parameter, and is ready to decide what to do with the alert, for example send it with a minimal action (e.g. ok), edit it, or cancel it. Processing takes place as though the user manually launched the application for creating the specified alert. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
243	O	Composing a process causes sending an operating system signal (see UNIX signaling) to the process with Process ID (PID) of the pid parameter (analogous to a Notify process action). A numeric value parameter (e.g. 0 or 1) may be communicated with the signal. An error can be logged if the process is not found for signaling. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
245	S	Composing a container causes launch of a MS container manager (e.g. file manager) to examine the contents of the container having the path in the container parameter (e.g. c:\dir1\subdir3) (analogous to a Notify container action). An error is logged if the file manager is not found for launch, or if the container is invalid. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
247	O	Composing a program object causes launch of a MS development environment application (e.g. Microsoft Visual Studio or IBM C-Set development consoles, etc), performing a search for the progobj parameter symbol, and producing search results of all occurrences for the current development working directory, mount point, or last used development repository. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.

Fig. 65B-6

Operand ↓	**PM**	**Preferred embodiment Compose processing**
249	**O**	Composing a cursor causes invocation of the appropriate application (e.g. graphic application by file type embodiment .bmp, .ico, etc) at the MS as though the user manually launched the application for the cursor parameter, and is ready for save of the file, or for further editing, or for cancellation. Depending on the cursor parameter referenced, an appropriate application will be launched for graphics, animation, etc. The cursor parameter is preferably a fully qualified path to determine the cursor (e.g. file). Processing takes place as though the user manually launched the application for the specified file. The embodiment's file type preference will influence which application is to be launched. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
251	**S**	Composing a calendar object causes interface to the calendar system for invoking the create calendar object interface and defaulting the appropriate calendar interface fields with the passed parameters. The calendar object parameter may be as described for Send calendar object, except for defaulting calendar interface create object interface(s). The user can subsequently create the scheduled event with little effort, or after optional modification, with a minimal action (e.g. click ok). Processing takes place as though the user manually launched the calendar application, entered the fields of the calendar form with passed parameters, and then is ready to decide what to do with it. Appropriate MS storage is updated and subsequently processed as though the user had manually started the calendar application and entered the calendar information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use. A standard calendaring interface is preferably used. Attributes can be set as is customary for a calendar object.
253	**S**	Composing an address book (AB) object causes interface to the AB system for invoking the create AB object interface and defaulting the appropriate AB interface fields with the passed parameters. The AB object parameter may be as described for Send AB object, except for defaulting AB interface create object interface(s). The user can subsequently create the AB entry with little effort, or after optional modification, with a minimal action (e.g. click ok). Processing takes place as though the user manually launched the AB application, entered the fields of the AB form with passed parameters, and then is ready to decide what to do with it. Appropriate MS storage is updated and subsequently processed as though the user manually started the AB application and entered the AB information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use. A standard AB interface is preferably used. Attributes can be set as may be customary for an AB entry.
. . .		

Fig. 65B-7

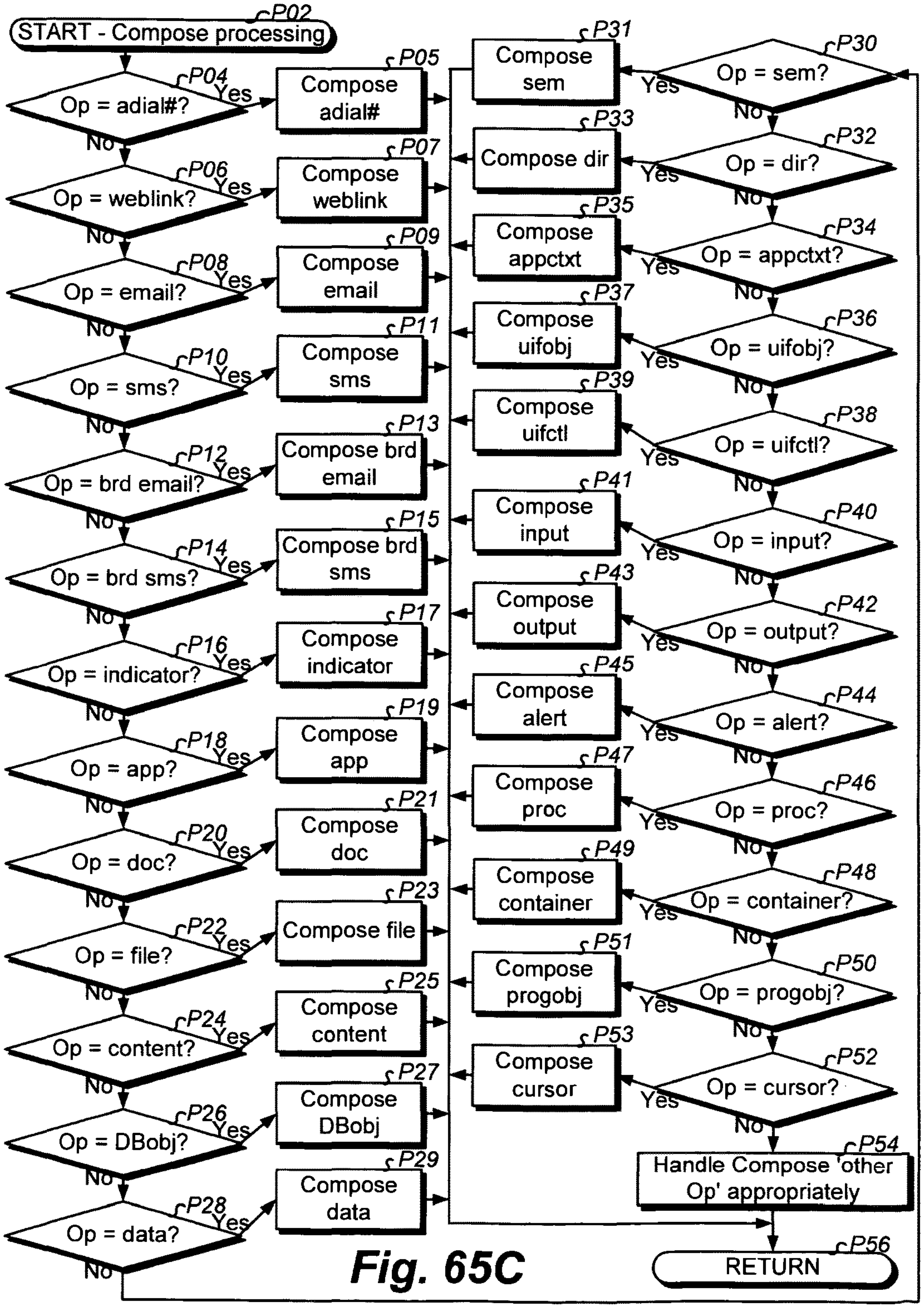
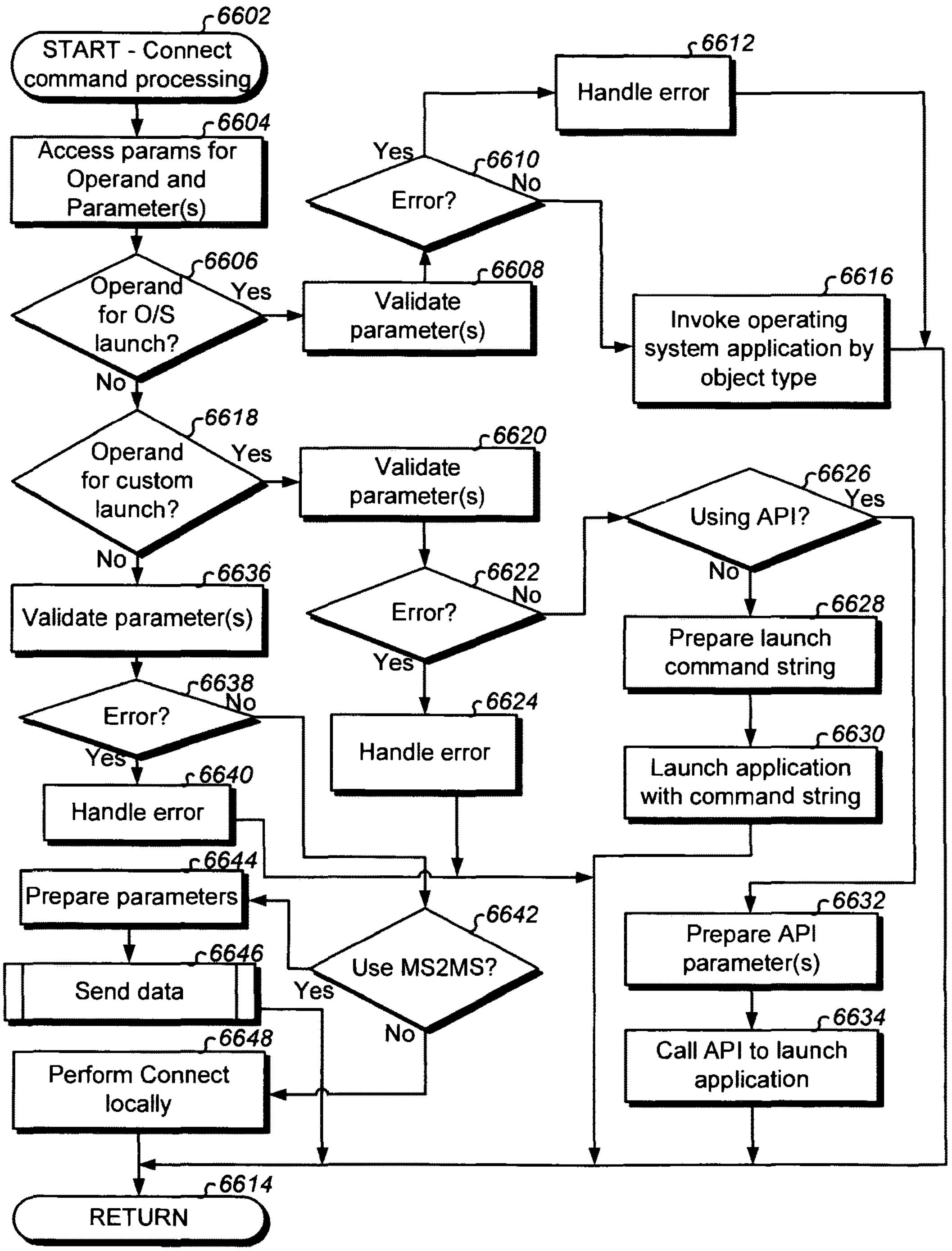

START - Compose processing
P02
Op = adial#?
P04
Yes
No
Compose adial#
P05
Op = weblink?
P06
Compose weblink
P07
Op = email?
P08
Compose email
P09
Op = sms?
P10
Compose sms
P11
Op = brd email?
P12
Compose brd email
P13
Op = brd sms?
P14
Compose brd sms
P15
Op = indicator?
P16
Compose indicator
P17
Op = app?
P18
Compose app
P19
Op = doc?
P20
Compose doc
P21
Op = file?
P22
Compose file
P23
Op = content?
P24
Compose content
P25
Op = DBobj?
P26
Compose DBobj
P27
Op = data?
P28
Compose data
P29
Op = sem?
P30
Compose sem
P31
Op = dir?
P32
Compose dir
P33
Op = appctxt?
P34
Compose appctxt
P35
Op = uifobj?
P36
Compose uifobj
P37
Op = uifctl?
P38
Compose uifctl
P39
Op = input?
P40
Compose input
P41
Op = output?
P42
Compose output
P43
Op = alert?
P44
Compose alert
P45
Op = proc?
P46
Compose proc
P47
Op = container?
P48
Compose container
P49
Op = progobj?
P50
Compose progobj
P51
Op = cursor?
P52
Compose cursor
P53
Handle Compose 'other Op' appropriately
P54
RETURN
P56

Fig. 65C



6602
START - Connect command processing
6604
Access params for Operand and Parameter(s)
6606
Operand for O/S launch?
Yes
No
6608
Validate parameter(s)
6610
Error?
Yes
No
6612
Handle error
6616
Invoke operating system application by object type
6618
Operand for custom launch?
Yes
No
6620
Validate parameter(s)
6622
Error?
No
Yes
6624
Handle error
6626
Using API?
Yes
No
6628
Prepare launch command string
6630
Launch application with command string
6632
Prepare API parameter(s)
6634
Call API to launch application
6636
Validate parameter(s)
6638
Error?
No
Yes
6640
Handle error
6642
Use MS2MS?
Yes
No
6644
Prepare parameters
6646
Send data
6648
Perform Connect locally
6614
RETURN

Fig. 66A

Operand ↓	PM	Preferred embodiment Connect processing
201	C	Connecting with an auto-dial # launches the MS phone number calling interface with the auto-dial # parameter defaulted for placing a call (like Notify autodial #). The call is actually made as though the user manually launched the dialing application, entered the auto-dial # and then chose to make the call with it. Appropriate MS storage is updated and subsequently processed as though the user had entered the # for calling manually and then made the call. Conventional call processing takes place thereafter. Preferably, the connect command data is maintained to LBX History, a historical call log (e.g. outgoing), or other useful storage for subsequent use.
203	S	Connecting with a web link launches a MS browser and invokes (transposes to) the link as though the user had entered it manually and went to the weblink page (like Notify weblink). In one embodiment, the weblink parameter includes URL parameter(s). In another embodiment, the params parameter supports a URL command string for appending to the weblink (e.g. "?v=yes&T=go") for customized web page processing. An alternate embodiment can fire form variables for active web page processing using the params parameter. Processing takes place as though the user manually launched the browser application, entered the weblink and then loaded the weblink webpage. Appropriate MS storage is updated and subsequently processed as though the user had entered the weblink in the browser manually. Preferably, the connect command data is maintained to LBX History, a historical log (web page load history), or other useful storage for subsequent use.
205	C	See Send Command for identical processing.
207	C	See Send Command for identical processing.
209	C	See Send Command for identical processing.
211	C	See Send Command for identical processing.
213	O	See Send Command for identical processing.
215	C	See Compose command for identical processing.
217	S	See Compose command for identical processing.
219	S	See Compose command for identical processing.
221	O	See Compose command for identical processing.
223	C	See Notify command for identical processing, except some applicable parameters not used.
225	O	See Compose command for identical processing.
227	O	See Compose command for identical processing.
229	O	See Compose command for identical processing.
231	C	See Compose command for identical processing.
233	S	See Compose command for identical processing.
235	O	See Compose command for identical processing.

Fig. 66B-1

Operand ↓	PM	Preferred embodiment Connect processing
237	O	See Compose command for identical processing.
239	O	See Compose command for identical processing.
241	C	Connecting with an alert causes interfacing to the alert subsystem for instantly producing the alert at the MS. Preferably, the connect command data is maintained to LBX History, a log, or other useful storage for subsequent use. The alert parameter of Notify processing is identical.
243	O	See Compose command for identical processing.
245	S	See Compose command for identical processing.
247	O	See Notify command for identical processing.
249	O	See Notify command for identical processing.
251	C	See Send Command for identical processing.
253	C	See Send Command for identical processing.
. . .		

Fig. 66B-2

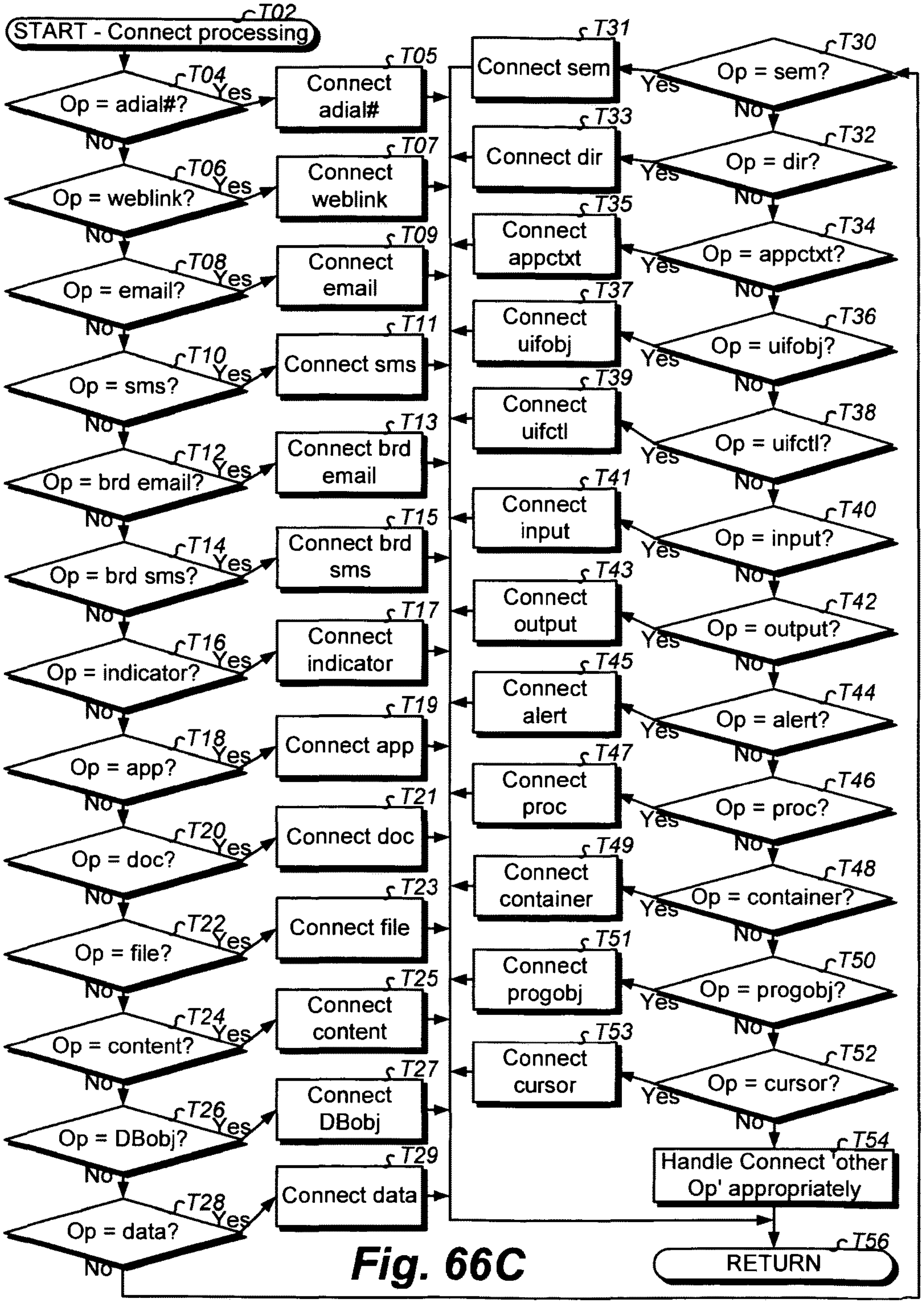

START - Connect processing
T02
Op = adial#?
T04
Yes
No
Connect adial#
T05
Op = weblink?
T06
Connect weblink
T07
Op = email?
T08
Connect email
T09
Op = sms?
T10
Connect sms
T11
Op = brd email?
T12
Connect brd email
T13
Op = brd sms?
T14
Connect brd sms
T15
Op = indicator?
T16
Connect indicator
T17
Op = app?
T18
Connect app
T19
Op = doc?
T20
Connect doc
T21
Op = file?
T22
Connect file
T23
Op = content?
T24
Connect content
T25
Op = DBobj?
T26
Connect DBobj
T27
Op = data?
T28
Connect data
T29
Op = sem?
T30
Connect sem
T31
Op = dir?
T32
Connect dir
T33
Op = appctxt?
T34
Connect appctxt
T35
Op = uifobj?
T36
Connect uifobj
T37
Op = uifctl?
T38
Connect uifctl
T39
Op = input?
T40
Connect input
T41
Op = output?
T42
Connect output
T43
Op = alert?
T44
Connect alert
T45
Op = proc?
T46
Connect proc
T47
Op = container?
T48
Connect container
T49
Op = progobj?
T50
Connect progobj
T51
Op = cursor?
T52
Connect cursor
T53
Handle Connect 'other Op' appropriately
T54
RETURN
T56

Fig. 66C

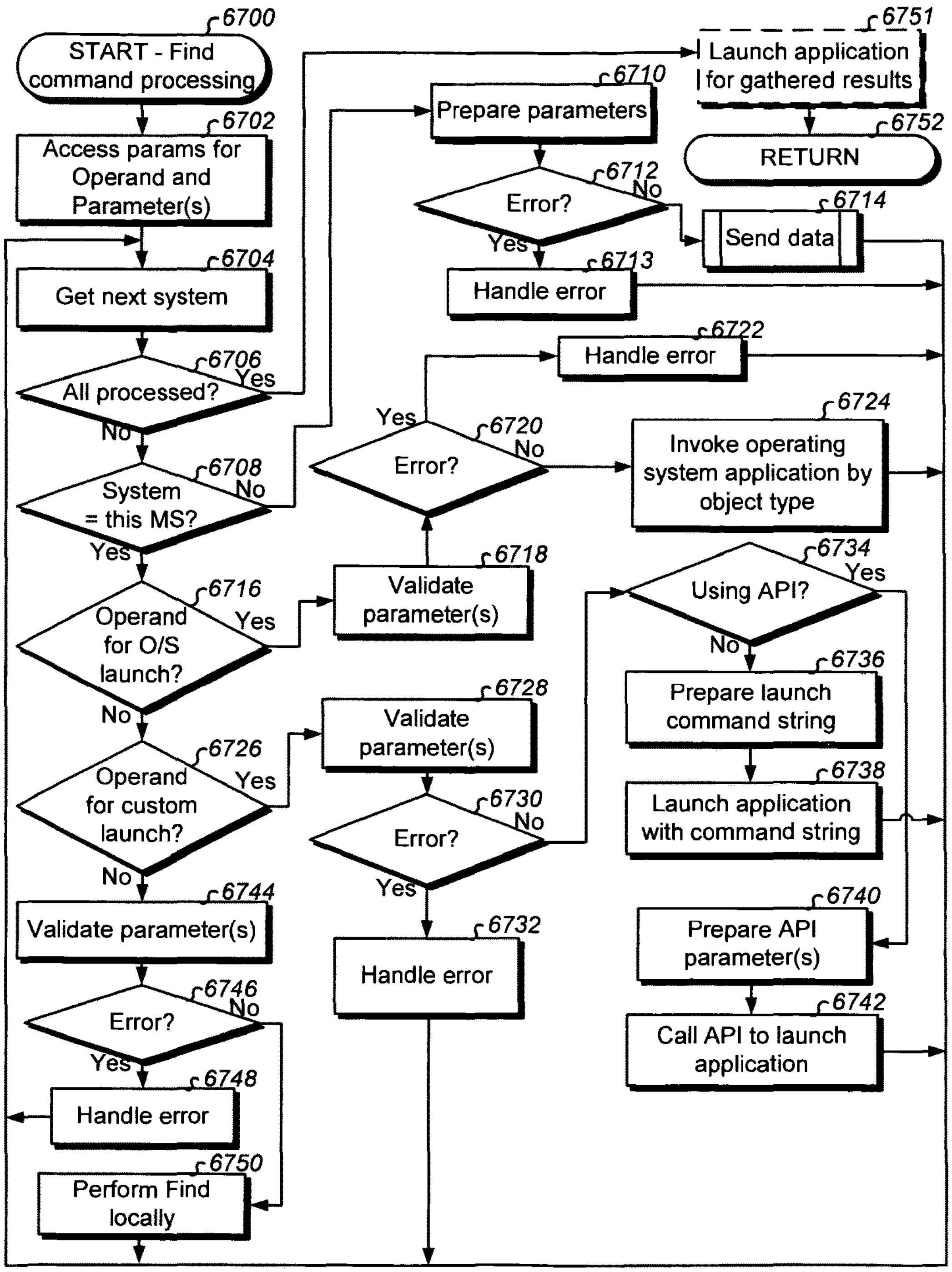

6700
START - Find command processing
6702
Access params for Operand and Parameter(s)
6704
Get next system
6706
All processed?
Yes
No
6708
System = this MS?
No
Yes
6716
Operand for O/S launch?
Yes
No
6726
Operand for custom launch?
Yes
No
6744
Validate parameter(s)
6746
Error?
No
Yes
6748
Handle error
6750
Perform Find locally
6710
Prepare parameters
6712
Error?
No
Yes
6713
Handle error
6722
Handle error
6720
Error?
Yes
No
6718
Validate parameter(s)
6728
Validate parameter(s)
6730
Error?
No
Yes
6732
Handle error
6751
Launch application for gathered results
6752
RETURN
6714
Send data
6724
Invoke operating system application by object type
6734
Using API?
Yes
No
6736
Prepare launch command string
6738
Launch application with command string
6740
Prepare API parameter(s)
6742
Call API to launch application

Fig. 67A

Operand ↓	**PM**	**Preferred embodiment Find processing**
201	**C**	Finding an auto-dial # launches a system (e.g. MS) phone number log interface with the auto-dial # parameter for searching. Preferably, both the outgoing and incoming logs are searched. The auto-dial # parameter can be a wildcard (pattern) for matching. In one embodiment, all matching occurrences found in history are presented with their date/time stamps, and perhaps other information, of the call and when it took place. In another embodiment, the most recent occurrence from a particular log is presented, and perhaps in an interface which enables calling the # with a minimal user action. The search takes place as though the user manually launched the search, entered the auto-dial # for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user manually performed the search. Preferably, the find cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. A new parameter can be specified for which log to search.
203	**S**	Finding a weblink launches a search to system (e.g. MS) browser history with the weblink parameter (and with the params parameter if specified) for searching. The weblink parameter can be a wildcard (pattern), and may include URL parameters, for matching. In one embodiment, all matching occurrences found in history are presented with their date/time stamps, and perhaps other information, of the link and when it was invoked. In another embodiment, the most recent occurrence from a particular invocation is presented, and perhaps in an interface which enables invoking (transposing to) the weblink with a minimal user action. In a preferred embodiment, the params parameter tells find processing how and where to search. The search takes place as though the user manually launched the search, entered the weblink for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. A new parameter can be specified for which folder to search.

Fig. 67B-1

Operand ↓	**PM**	**Preferred embodiment Find processing**
205	**C**	Finding an email causes searching a system (e.g. MS) email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding (patterns for matching). Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel';recip:'george@alltell.com';body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by seaching for substrings. All occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS email system processing from that point forward (e.g. when processed at local MS). Alternatively, an additional parameter indicates how to search. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-2

Operand ↓	PM	Preferred embodiment Find processing
207	C	Finding an sms message causes searching a system (e.g. MS) sms messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In the preferred embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS messaging system processing from that point forward (e.g. when processed at local MS). Alternatively, an additional parameter indicates how to search. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-3

Operand ↓	**PM**	**Preferred embodiment Find processing**
209	**C**	Finding a broadcast email causes searching a system (e.g. MS) email system with search criteria of the email param string. The email param string can specify searching any email fields for any values including wildcarding. Each field is referenced with a predefined name and associated with a search criteria. For example, the param of "subj:'personnel';recip:'george@alltell.com';body:'reduction in force'" searches emails with a subject containing "personnel" and sent to "george@alltell.com" and has an email body containing "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In the preferred embodiment, all occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other info, of the email and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS email system processing from that point forward (e.g. when processed at local MS). Alternatively, an additional parameter indicates how to search. The search takes place as though the user manually launched the search, entered criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-4

Operand ↓	PM	Preferred embodiment Find processing
211	C	Finding a broadcast sms message causes searching a system (e.g. MS) sms messaging system with search criteria of the sms message param string. The message param string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com';" causes searching messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In the preferred embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS messaging system processing from that point forward (e.g. when processed at local MS). Alternatively, an additional parameter indicates how to search. The search takes place as though the user manually launched the search, entered search criteria, and then was presented with result(s). Appropriate MS storage is updated and processed as though the user manually performed the search. Preferably, find cmd data is maintained to LBX History, a historical log, or other useful storage for later use.
213	O	Finding an indicator searches appropriate system (e.g. MS) storage for the indicator (e.g. storage/memory used for indicators by other commands). The indicator parameter string specifies the indicator (e.g. string) being sought and wildcarding is supported. Any active user interface object containing the indicator is surfaced. If more than one user interface object contains the indicator, then all objects are appropriately tiled with the most recent in the priority position(s). In another embodiment, appropriate MS storage/memory which contains the history of indicators sent is searched and all occurrences found in history are presented with at least their date/time stamps, the indicator, and perhaps other information. In yet another embodiment, a new parameter tells processing whether to surface/prioritize active objects, or to search history for when indicators were sent. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-5

Operand ↓	**PM**	**Preferred embodiment Find processing**
215	**C**	Finding an application causes searching the system (e.g. MS) for the application (and with the params parameter if specified). The app parameter is preferably an executable name, optionally with parameters that were passed. Providing a partial or full path to the application parameter will validate that it is found there. The app parameter string preferably supports wildcarding. Embodiments (or as specified with params and/or new parameters) include file system searching, invocation history (e.g. Microsoft Windows up/down arrow command line recall) searching for what had been invoked (perhaps within a trailing time period), what is currently running, what has been terminated (perhaps within a trailing time period), or any of these for a particular invoked identity, credentials, or owner, In the preferred embodiment, all occurrences found on the MS and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information. In another embodiment, all parts which are linked to the executable are identified with their paths, date/time stamps, size, and perhaps attributes when a symbol file is specified with a new parameter. The symbol file is output from a link process and can be used to identify all executable parts such as dynamic link libraries, linked binaries, and any other executable binary file involved with the application. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and processed as though the user had manually performed the search. Preferably, find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
217	**S**	Finding a document causes searching the system (e.g. MS) for the document. The doc parameter is a document name. The document parameter can be a wildcard (pattern) for matching. Providing a partial or full path to the document name will validate that it is found there. In the preferred embodiment, all occurrences found on the MS and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
219	**S**	Finding a file causes searching the system (e.g. MS) for the file. The path parameter is a file name. Providing a partial or full path to the file will validate that it is found there. The path parameter can be a wildcard (pattern) for matching. In the preferred embodiment, all occurrences found on the MS and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for

Fig. 67B-6

Operand ↓	**PM**	**Preferred embodiment Find processing**
		subsequent use.
221	**O**	Finding content causes searching the system (e.g. MS) for the content. The content parameter can be a wildcard (pattern) for matching. The content parameter can be a handle to the content, or a search criteria for the content. In the preferred embodiment, all occurrences found on the MS and where the content is located is presented to the user, perhaps with other content description information. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). There are various embodiments for how and where content is maintained. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
223	**C**	Finding a DB object causes searching the system (e.g. MS) database(s) for the database object. The database object parameter is provided with a variety of syntaxes depending on the type of database object sought. For example, the DB-obj parameters is "T:tablename" to seek a table, "S:schemaname" to seek a particular schema, "C:columnname" to seek a particular column name, "D:DBname" to seek a particular DB name, "R:rolename" to seek a particular role set, "P:procname" to search for particular stored procedure, etc. There are unique syntaxes for every type of DB object being sought. Those skilled in the art know how to query system tables for particular DB object(s) sought. An appropriate SQL client API should be used. If necessary, an additional parameter is specified for authentication credentials (may be specified with DB-obj string syntax). The search criteria can be a wildcard (pattern) for matching. In the preferred embodiment, all occurrences found on the MS and information about the occurrence is presented to the user. The search takes place as though the user manually launched the search, entered the criteria or query for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-7

Operand ↓	PM	Preferred embodiment Find processing
225	O	Finding data causes searching the system (e.g. MS) for the data. In the preferred embodiment, the data is a global system variable visible to processes of a MS O/S. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). Depending on the embodiment, data may be that which is contained in a program data segment, stack segment, and/or extra segment. There can be unique syntaxes for specifying which type of data is being sought (e.g. "S:dataname"). The search criteria can be a wildcard (pattern) for matching. In the preferred embodiment, all occurrences found on the MS and information about the occurrence including its current value is presented to the user. In one embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. An AppTerm uses record 5300 for access. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, find command data is maintained to LBX History, a historical log, or other useful storage for later use.
227	O	Finding a semaphore causes reading the current value of the semaphore at the system (e.g. MS) where the find command action is being executed and then presenting the current value along with any other useful information for the semaphore. The semaphore param can be a wildcard (pattern) for matching. In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
229	S	Finding a directory causes searching the system (e.g. MS) for the directory (path). The path parameter is a directory name. Providing a more defined partial or full path to the path parameter will narrow down the results if the directory exists in more than one place. The path parameter can be a wildcard (pattern) for matching. In the preferred embodiment, all occurrences found on the MS and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-8

Operand ↓	**PM**	**Preferred embodiment Find processing**
231	**C**	Finding an application context causes invocation of the application at the system (e.g. MS) and then executing a macro within the application context (similar to Compose app object processing). The app parameter is preferably a fully qualified path name to the executable to start. In another embodiment, the app parameter is indirect: a path name to a "shortcut" (like a MS Windows shortcut). In another embodiment, the app parameter is an identifier string for the underlying operating system to know which application to start. The search criteria can be a wildcard (pattern) for matching. The macro parameter is preferably a file, or path, or accessible variable name containing a set of keystrokes that can be directed to standard/user-interface input. In another embodiment, the macro parameter is a prerecorded user input scenario (for play after application launched -- pulldown selections, mouse droppings, clicks, etc) captured to a file or stored in an accessible variable name. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
233	**S**	Finding a user interface object causes finding and focusing the user interface object at the system (e.g. MS) which contains the object text (objtxt) parameter. In a preferred embodiment, there is a unique syntax for which places of user interface objects that are currently active are to be search (e.g. title bar, entry fields, radio button options, window text, combinations thereof, etc). The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
235	**O**	Finding user interface control causes searching the system (e.g. MS) storage and/or memory which was used for processing another command (e.g. Compose) to redirect the keystroke macro to standard input of the MS as if it were entered by the MS user. The search criteria can be a wildcard (pattern) for matching. The macro parameter is the same as was used by the command and is to be matched. In the preferred embodiment, presented in the search results are all occurrences of previous command actions, which used the macro at the MS, including the command, date/time stamp, and other information recorded (e.g. to LBX History, a historical log, or other useful storage for subsequent use). The search takes place as though the user manually launched a search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, historical log, or other useful storage for subsequent use.

Fig. 67B-9

Operand ↓	**PM**	**Preferred embodiment Find processing**
237	**O**	Finding input causes searching the system (e.g. MS) storage and/or memory which was used for processing another command (e.g. Compose) to redirect the input to the iodev device of the MS. The iodev and input parameters are the same as was used by a previous command and is to be matched. The search criteria can be a wildcard (pattern) for matching. In the preferred embodiment, presented in the search results are all occurrences of previous command actions, which used the iodev and input at the MS, including the command, date/time stamp, and other information recorded (e.g. to LBX History, a historical log, or other useful storage for subsequent use). The search takes place as though the user manually launched a search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
239	**O**	Finding output causes searching the system (e.g. MS) storage and/or memory which was used for processing another command (e.g. Compose) to redirect the output to the iodev device of the MS. The search criteria can be a wildcard (pattern) for matching. The iodev and output parameters are the same as was used by a previous command and is to be matched. In the preferred embodiment, presented in the search results are all occurrences of previous command actions, which used the iodev and output at the MS, including the command, date/time stamp, and other information recorded (e.g. to LBX History, a historical log, or other useful storage for subsequent use). The search takes place as though the user manually launched a search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
241	**S**	Finding an alert causes searching the system (e.g. MS) for the alert. The alert parameter is the same parameter used to generate an alert (e.g. using another command). In the preferred embodiment, all occurrences found on the MS which is associated to the alerter application in use at the MS, and which is used for other commands disclosed, are presented to the user with at least their date/time stamps, and perhaps other information. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). The search criteria can be a wildcard (pattern) for matching. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-10

Operand ↓	PM	Preferred embodiment Find processing
243	O	Finding a process causes finding all process names running at the system (e.g. MS) which contain the prname string parameter (e.g. in UNIX: "ps -ef | grep prname"). In the preferred embodiment, all occurrences found running at the MS are presented with interesting programmatic information such as when started, its size, etc (see UNIX ps command for other information that can be presented here in various embodiments; an additional parameter (like ps parameters) can specify what info to provide to the user). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, historical log, or other useful storage for subsequent use.
245	S	Finding a container causes searching the system (e.g. MS) for the container. The container parameter is a container name (e.g. file system directory) depending on the MS or environment. Unique syntaxes can be used for which type of container is being searched. In the preferred embodiment, all occurrences found on the MS and their paths are presented to the user with information of interest. The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, historical log, or other useful storage for subsequent use.
247	O	Finding a program object causes searching the system (e.g. MS) for the program object. In the preferred embodiment, a unique syntax is used for which type of program object is being sought (e.g. Q:queuename has a queue qualifier prefix). There can be unique syntaxes for specifying which type of program object is being sought (e.g. "S:dataname"). In the preferred embodiment, all occurrences found on the MS and information about the occurrence including its current value is presented to the user. A null data parameter returns all occurrences found. A non-null data parameter returns these objects having the data value. Objects must be programmatically accessible. In one embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. In another embodiment, MS storage and/or memory is searched which recorded a previous atomic command action, and the search takes place for a previous command(s) (e.g. Notify) for when performed and what was performed. The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-11

Operand ↓	PM	Preferred embodiment Find processing
249	O	Finding a cursor causes searching the system (e.g. MS) storage and/or memory which was used for processing another command (e.g. Compose) to view or alter a cursor. In the preferred embodiment, presented in the search results are all occurrences of previous command actions, which viewed or altered the cursor at the MS, including the command, date/time stamp, and other information recorded (e.g. to LBX History, a historical log, or other useful storage for subsequent use). The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched a search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
251	C	Finding a calendar object causes searching a system (e.g. MS) calendar system with search criteria of the calendar object parameter string. The calendar object parameter string can specify searching any calendar entry fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria (similar to email above). Those skilled in the art recognize useful syntaxes for searching any characteristics of calendar objects with an appropriate syntax. The calendar object parameter is at least a string with a syntax for querying any combination of calendar object fields. In the preferred embodiment, all occurrences found in history are presented with at least their date/time stamps, attendees, and perhaps other information, of the calendar object and when it was scheduled. In another embodiment, the most recent occurrence from the calendaring system is presented, and perhaps in an interface which enables appropriate MS calendar system processing from that point forward (e.g. when processed at local MS), or alternatively an additional parameter can specify how to search. Wildcarding (pattern matching) is preferably inherent by searching for substrings. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-12

Operand ↓	**PM**	**Preferred embodiment Find processing**
253	**C**	Finding an address book (AB) object causes searching a system (e.g. MS) AB system with search criteria of the AB object parameter string. The AB object parameter string can specify searching any AB entry fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria (similar to email above). Those skilled in the art recognize many useful syntaxes for searching any characteristics of AB objects/entries with an appropriate syntax. The AB object parameter is at least a string with a syntax for querying any combination of AB object fields. In the preferred embodiment, all occurrences found are presented with appropriate AB information. In another embodiment, the most recent occurrence from the AB system is presented, and perhaps in an interface which enables appropriate MS AB system processing from that point forward (e.g. when processed at local MS), or alternatively an additional parameter can specify how to search. Wildcarding (pattern matching) is preferably inherent by searching for substrings. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
. . .		

Fig. 67B-13

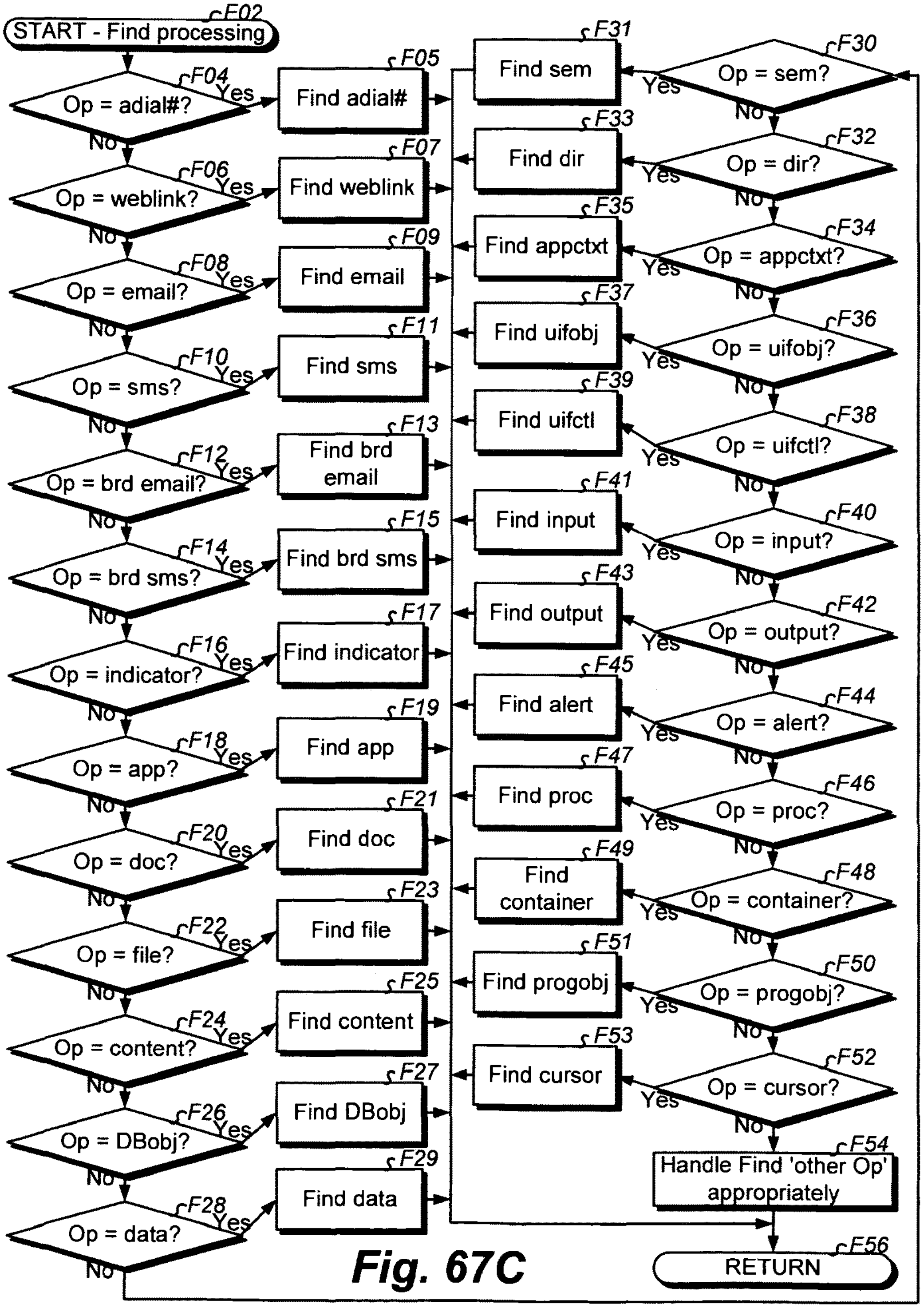

START - Find processing
F02
Op = adial#?
F04
Yes
No
Find adial#
F05
Op = weblink?
F06
Find weblink
F07
Op = email?
F08
Find email
F09
Op = sms?
F10
Find sms
F11
Op = brd email?
F12
Find brd email
F13
Op = brd sms?
F14
Find brd sms
F15
Op = indicator?
F16
Find indicator
F17
Op = app?
F18
Find app
F19
Op = doc?
F20
Find doc
F21
Op = file?
F22
Find file
F23
Op = content?
F24
Find content
F25
Op = DBobj?
F26
Find DBobj
F27
Op = data?
F28
Find data
F29
Op = sem?
F30
Find sem
F31
Op = dir?
F32
Find dir
F33
Op = appctxt?
F34
Find appctxt
F35
Op = uifobj?
F36
Find uifobj
F37
Op = uifctl?
F38
Find uifctl
F39
Op = input?
F40
Find input
F41
Op = output?
F42
Find output
F43
Op = alert?
F44
Find alert
F45
Op = proc?
F46
Find proc
F47
Op = container?
F48
Find container
F49
Op = progobj?
F50
Find progobj
F51
Op = cursor?
F52
Find cursor
F53
Handle Find 'other Op' appropriately
F54
RETURN
F56

Fig. 67C

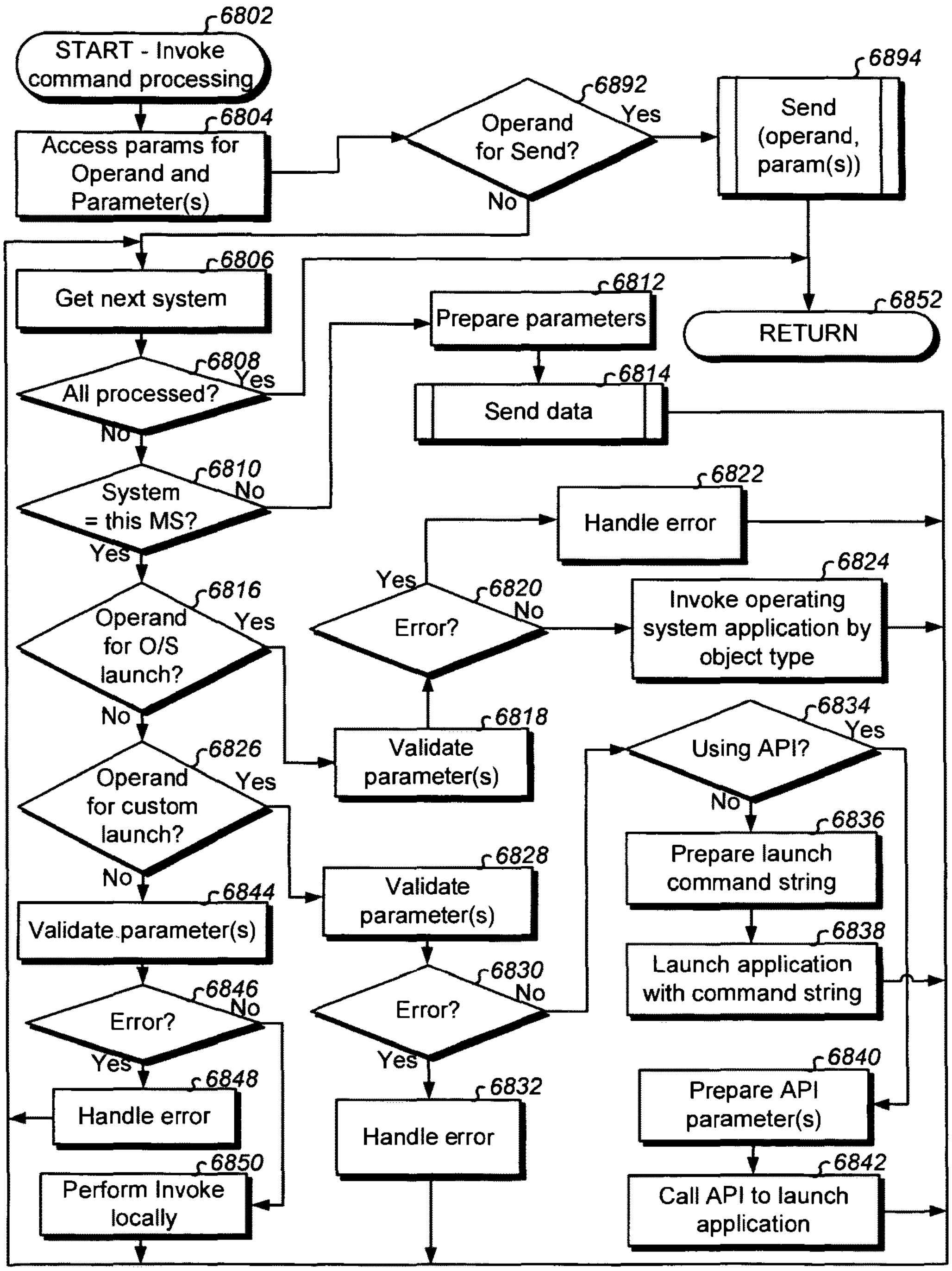

6802
START - Invoke command processing
6804
Access params for Operand and Parameter(s)
6892
Operand for Send?
Yes
No
6894
Send (operand, param(s))
6806
Get next system
6852
RETURN
6812
Prepare parameters
6808
All processed?
Yes
No
6814
Send data
6810
System = this MS?
No
Yes
6822
Handle error
6816
Operand for O/S launch?
Yes
No
6820
Error?
Yes
No
6824
Invoke operating system application by object type
6818
Validate parameter(s)
6834
Using API?
Yes
No
6826
Operand for custom launch?
Yes
No
6836
Prepare launch command string
6844
Validate parameter(s)
6828
Validate parameter(s)
6838
Launch application with command string
6846
Error?
No
Yes
6830
Error?
No
Yes
6840
Prepare API parameter(s)
6848
Handle error
6832
Handle error
6850
Perform Invoke locally
6842
Call API to launch application

Fig. 68A

Operand ↓	PM	Preferred embodiment Invoke processing
201	C	Invoking with an auto-dial # launches the MS phone number calling interface at the MS (local or remote) with the auto-dial # parameter for placing a call (like/see Notify / Connect autodial # processing). The call is actually made as though a user manually launched the dialing application at the particular MS, entered the auto-dial # and then chose to make the call with it. Appropriate MS storage is updated and subsequently processed as though the user had entered the # for calling manually, and then make the call. Conventional call processing takes place thereafter. Preferably, the invoke command data is maintained to LBX History, a historical call log (e.g. outgoing), or other useful storage for subsequent use. A system parameter provides means for placing the call from another system (e.g. another MS) – like a Host specification.
203	S	Invoking with a web link launches the MS browser at the particular MS and invokes (transposes to) the link as though the user had entered it manually and went to the weblink page (like/see Notify / Connect weblink processing). In one embodiment, the weblink parameter includes URL parameter(s). In another embodiment, the params parameter supports a URL command string for appending to the weblink (e.g. "?v=yes&T=go") for customized web page processing. An alternate embodiment can fire form variables for active web page processing using the params parameter, or URL parameter(s). Processing takes place as though the user manually launched the browser application at the particular MS, entered the weblink and then loaded the weblink webpage. Appropriate MS storage is updated and subsequently processed as though the user had entered and invoked the weblink in the browser manually. Preferably, the compose command data is maintained to LBX History, a historical log (web page load history), or other useful storage for subsequent use.
205	E	See Send Command for identical processing.
207	E	See Send Command for identical processing.
209	E	See Send Command for identical processing.
211	E	See Send Command for identical processing.

Fig. 68B-1

Operand ↓	PM	Preferred embodiment Invoke processing
213	O	Invoking an indicator updates the appropriate MS storage so that the currently focused user interface object (e.g. window titlebar) of the particular MS user interface is modified with the indicator (like/see Send indicator processing). If there are no active user interface objects in the MS user interface, then an appropriate alert area of the currently focused interface is to display the indicator. The user can clear (remove) the indicator when desired. Preferably, the indicator is used for modifying other focused objects (e.g. titlebars) or other focused areas in the user interface so as to not get overlooked. For example, as the user navigates and surfaces/focuses new user interface objects, the indicator remains visible on the newly focused object. Preferably, the indicator is selectable by the user of the MS for showing all other send command parameters associated, as well as a date/time stamp of when sent. In other embodiments, the most recently displayed indicator is displayed in the appropriate focused area, but the user can conveniently select any indicators which were sent in history at some point in time for sought indicator information by selecting the currently displayed indicator and then requesting to browse/scroll history of previously delivered indicators (with options to see details). Preferably, the invoke command data is maintained to LBX History, a historical log (web page load history), or other useful storage for subsequent use.
215	C	Invoking an application causes invocation of the application at the particular MS (like/see Send app processing). The app parameter is preferably a fully qualified path name to the executable to start, and may already include parameters. In another embodiment, the app parameter is indirect: a path name to a "shortcut" (like a MS Windows shortcut). In another embodiment, the app parameter is an identifier string for the underlying operating system to know which application to start. The params parameter may specify the executable parameters, or may be used for how to start the application (like attributes of Send app processing). An error is logged if the app parameter is not found for launch. Preferably, the invoke command data is maintained to LBX History, a historical log (web page load history), or other useful storage for subsequent use.
217	S	Invoking a document causes invocation of an appropriate application at the particular MS in accordance with the object type as though the user selected the document for automatically being associated to the correct application when opening the document. The user can then decide what to do with the document once it is opened in the appropriate application. In an alternate embodiment, an additional parameter is provided for exactly what to do with the document, in which case an appropriate API is invoked with the document (i.e. PM = C). The doc parameter is preferably a fully qualified path name to the document. Preferably, the invoke command data is maintained to LBX History, a log, or other useful storage for subsequent use.

Fig. 68B-2

Operand ↓	PM	Preferred embodiment Invoke processing
219	S	Invoking a file causes invocation of an appropriate application at the particular MS in accordance with the file type as though the user selected the file for automatically being associated to the correct application when opening the document. Processing takes place as though the user manually launched the application for the specified file. The user can then decide what to do with the file once it is opened in the appropriate application. In an alternate embodiment, an additional parameter is provided for exactly what to do with the file, in which case an appropriate API is invoked with the file (i.e. PM = C). The path parameter is preferably a fully qualified path name to the file. Preferably, the invoke command data is maintained to LBX History, a log, or other useful storage for subsequent use.
221	O	Invoking content causes the content to be presented at the particular MS in a manner which is appropriate for the content type (like/see Notify content processing). The content parameter is one that cannot be classified in the other operands, but is content for presentation nevertheless. Examples include special data records (e.g. extern variable name), content data memory locations (e.g. programmatic variable), or files containing a customizably processed format. Methods of displaying the content include audio and/or visual using applicable MS capabilities. Preferably, the invoke command data is maintained to LBX History, a historical content log (e.g. incoming), browser history data, or other useful storage for subsequent user browse of the accompanying content and a date/time stamp of when sent, and for presentation of the content. Various embodiments will save to LBX History how many times, and when, the content was presented.
223	O	Invoking a Database (DB) object causes the DB object (i.e. qualified database with access query string) to be modified with the query parameter at the particular MS (like/see Notify DB-obj processing without certain parameters). The query parameter is used to perform any query against the specified DB-database (DB-obj), preferably a query that only returns a return code. Fig. 75B processing may return SELECT results in some embodiments (like find results returned). Preferably, the invoke command data is maintained to LBX History, a database log (e.g. incoming), or other useful storage for subsequent query use and a date/time stamp of when sent, and for DB query manager browse/use of the query in response to an applicable user action.
225	O	Invoking data causes modifying the value of the data at the particular MS (i.e. set data to value – like/see Notify data processing without certain parameters). An error can result if the data is not resolvable for the attempt. In the preferred embodiment, the data is a global system variable visible to all processes of a MS operating system. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). A recognized AppTerm causes access to record 5300 for proper semaphore synchronized access. Preferably, the data affected is maintained to LBX History, a historical log (e.g. incoming), or other useful storage for subsequent user browse, or programmatic access, of the data variable name, its before and after values & date/time stamp of sent, and for presentation of the data value for a user action to show it.

Fig. 68B-3

Operand ↓	PM	Preferred embodiment Invoke processing
227	O	Invoking a semaphore causes modifying the value of the semaphore at the particular MS where the action is being executed (like/see Notify semaphore processing). In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing (e.g. RAM semaphore). Preferably, the semaphore value before and after setting is maintained to LBX History, a historical log (e.g. incoming), or other useful storage for subsequent user browse, or programmatic access, and for presentation of the semaphore information in response to a user action to show it.
229	S	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
231	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
233	S	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter). Also, the cmds parameter replaces the "capture focused object" command string with one or more semicolon delimited "capture focused object" command strings for each target system in the system(s) parameters. This enables a plurality of different types of MSs to participate even though they have different commands (e.g. keystroke capture actions) to accomplish capturing the focused user interface object. Based on the file type at the particular MS, the appropriate application opens the file.
235	O	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
237	O	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
239	O	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
241	C	See Connect Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
243	O	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
245	S	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
247	O	See Notify Command for identical processing, except sender is forced to requesting MS, no documentary subj/msg parameter, and system(s) used instead of recipient(s). Processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).

Fig. 68B-4

Operand ↓	**PM**	**Preferred embodiment Invoke processing**
249	**O**	See Notify Command for identical processing, except sender is forced to requesting MS, no documentary subj/msg parameter, and system(s) used instead of recipient(s). Processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
251	**O**	See Send Command for identical processing, except sender is forced to requesting MS, and system(s) used instead of recipient(s) for calendar alteration without regard for the owner (API embodiment). Processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
253	**O**	See Send Command for identical processing, except sender is forced to requesting MS, and system(s) used instead of recipient(s) for AB alteration without regard for the owner (API embodiment). Processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
. . .		

Fig. 68B-5

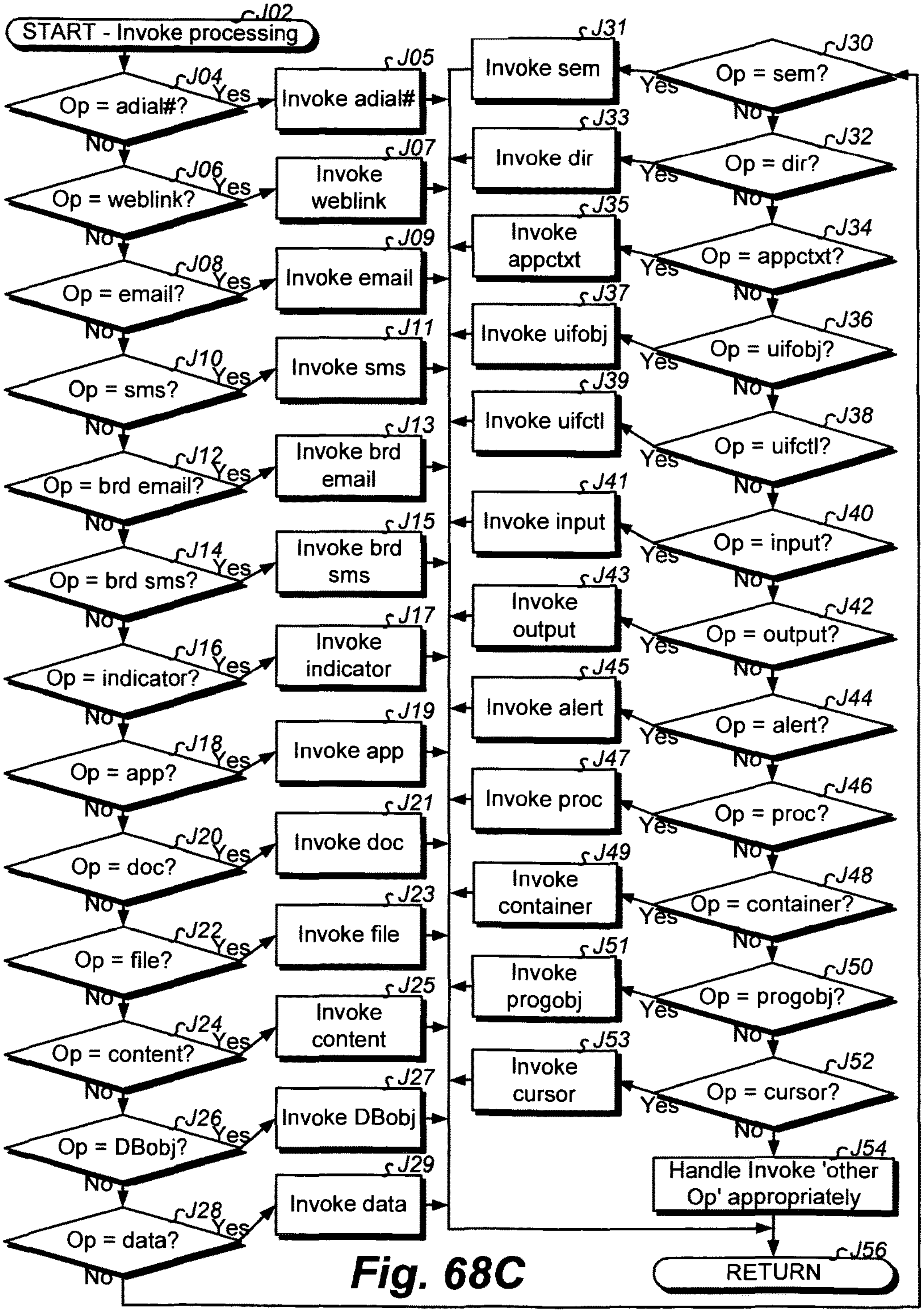

START - Invoke processing
J02
Op = adial#?
J04
Yes
No
Invoke adial#
J05
Op = weblink?
J06
Invoke weblink
J07
Op = email?
J08
Invoke email
J09
Op = sms?
J10
Invoke sms
J11
Op = brd email?
J12
Invoke brd email
J13
Op = brd sms?
J14
Invoke brd sms
J15
Op = indicator?
J16
Invoke indicator
J17
Op = app?
J18
Invoke app
J19
Op = doc?
J20
Invoke doc
J21
Op = file?
J22
Invoke file
J23
Op = content?
J24
Invoke content
J25
Op = DBobj?
J26
Invoke DBobj
J27
Op = data?
J28
Invoke data
J29
Op = sem?
J30
Invoke sem
J31
Op = dir?
J32
Invoke dir
J33
Op = appctxt?
J34
Invoke appctxt
J35
Op = uifobj?
J36
Invoke uifobj
J37
Op = uifctl?
J38
Invoke uifctl
J39
Op = input?
J40
Invoke input
J41
Op = output?
J42
Invoke output
J43
Op = alert?
J44
Invoke alert
J45
Op = proc?
J46
Invoke proc
J47
Op = container?
J48
Invoke container
J49
Op = progobj?
J50
Invoke progobj
J51
Op = cursor?
J52
Invoke cursor
J53
Handle Invoke 'other Op' appropriately
J54
RETURN
J56

Fig. 68C

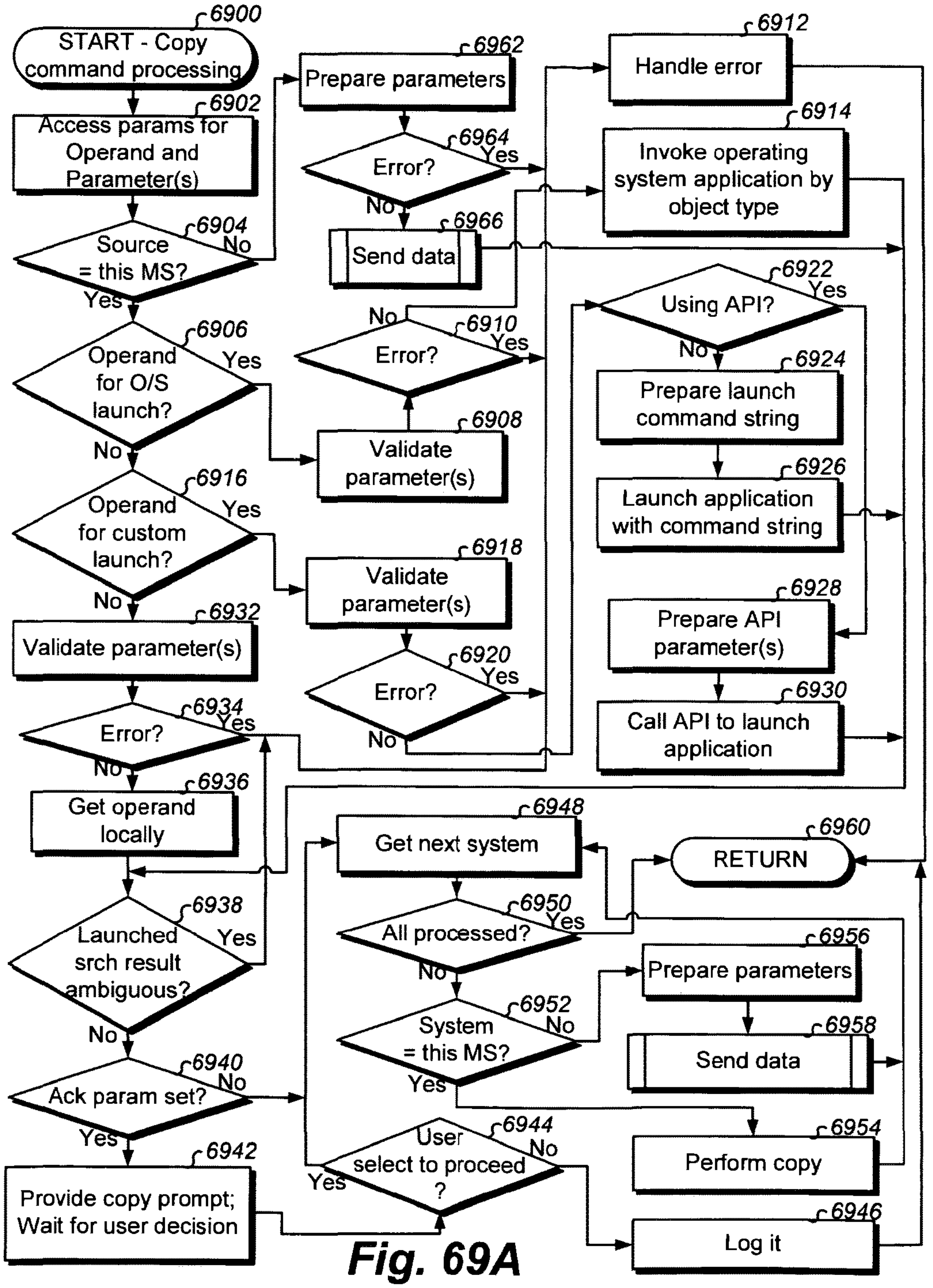

6900
START - Copy command processing
6902
Access params for Operand and Parameter(s)
6904
Source = this MS?
No
Yes
6906
Operand for O/S launch?
Yes
No
6916
Operand for custom launch?
Yes
No
6932
Validate parameter(s)
6934
Error?
Yes
No
6936
Get operand locally
6938
Launched srch result ambiguous?
Yes
No
6940
Ack param set?
No
Yes
6942
Provide copy prompt; Wait for user decision
6962
Prepare parameters
6964
Error?
Yes
No
6966
Send data
No
6910
Error?
Yes
6908
Validate parameter(s)
6918
Validate parameter(s)
6920
Error?
Yes
No
6912
Handle error
6914
Invoke operating system application by object type
6922
Using API?
Yes
No
6924
Prepare launch command string
6926
Launch application with command string
6928
Prepare API parameter(s)
6930
Call API to launch application
6948
Get next system
6960
RETURN
6950
All processed?
Yes
No
6956
Prepare parameters
6952
System = this MS?
No
Yes
6958
Send data
6954
Perform copy
6944
User select to proceed ?
No
Yes
6946
Log it

Fig. 69A

Operand ↓	PM	Preferred embodiment Copy processing
201	C	Copying an auto-dial # launches a phone number log interface with the auto-dial # parameter (can be wildcarded) for searching the source system. Preferably, both the outgoing and incoming logs are searched. In an alternate embodiment, the log is specified with a parameter. In the preferred embodiment, the most recent occurrence from a particular log is provided. In another embodiment, all occurrences found in history are presented with their date/time stamps, and perhaps other information, of the call and when it took place (e.g. when the ack parameter is set) and the user browses the results prior to accepting the copy of multiple items. The search takes place as though the user manually launched the search, entered the auto-dial # for the search, and then was provided with result(s) for the copy. Preferably, the copy shall take place if there are no ambiguities (e.g. more than one phone number returned per search criteria). An additional parameter may be specified for the target (different log) of the copy, otherwise the object is copied to an assumed location (e.g. same folder to more recent position). Appropriate MS storage is updated and subsequently processed as though the user manually performed the search and copy of the result. Preferably, the copy cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made to system(s) logs, preferably with identifying information of the source and who did the copy.
203	C	Copying a weblink launches a search to MS browser history with the weblink parameter (can be wildcarded) for searching the source system. In one embodiment, all occurrences found in history are presented with their date/time stamps, and perhaps other information, of the link and when it was invoked (e.g. when the ack parameter is specified to true) for presentation to the user prior to doing the copy. In the preferred embodiment, the most recent occurrence from a particular invocation is provided for the copy. An additional parameter may be specified for the target (specified favorites folder), otherwise the object is copied to an assumed location (e.g. highest level favorites folder or special named folder).The search takes place as though the user manually launched the search, entered the weblink for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy of the result. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made to a special browser favorites folder, or another designated folder configured ahead of time, preferably with identifying information of the source and who did the copy.

Fig. 69B-1

Operand ↓	**PM**	**Preferred embodiment Copy processing**
205	**C**	Copying an email causes searching the source email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding (patterns for matching). Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel';recip:'george@alltell.com';body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for the copy. An additional parameter may be specified for the target (specified folder), otherwise the object is copied to an assumed location (e.g. drafts, inbox, or special named folder). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for doing the copy. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made to a special email folder of the target system, or another designated folder configured ahead of time, or as specified with a new parameter for copy processing, preferably with identifying information of the source and who did the copy (if supported in email application).

Fig. 69B-2

Operand ↓	**PM**	**Preferred embodiment Copy processing**
207	**C**	Copying an sms message causes searching the source messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for copying. An additional parameter may be specified for the target (specified folder), otherwise the object is copied to an assumed location (e.g. inbox, drafts, special named folder). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made to a special messaging folder of the target system, or another designated folder configured ahead of time, or as specified with a new parameter to copy processing, preferably with identifying information of the source and who did the copy.

Fig. 69B-3

Operand ↓	**PM**	**Preferred embodiment Copy processing**
209	**C**	Copying a broadcast email causes searching the source email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding (patterns for matching). Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel';recip:'george@alltell.com';body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for the copy. An additional parameter may be specified for the target (specified folder), otherwise the object is copied to an assumed location (e.g. inbox, drafts, special named folder). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for doing the copy. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made to a special email folder of the target system, or another designated folder configured ahead of time, or as specified with a new parameter to copy processing, preferably with identifying information of the source and who did the copy.

Fig. 69B-4

Operand ↓	**PM**	**Preferred embodiment Copy processing**
211	**C**	Copying a broadcast sms msg causes searching the source messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for copying. An additional parameter may be specified for the target (specified folder), otherwise the object is copied to an assumed location (e.g. inbox, drafts, special named folder). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made to a special messaging folder of the target system, or another designated folder configured ahead of time, or as specified with a new parameter to copy processing, preferably with identifying information of the source and who did the copy.
213	**C**	Copying an indicator searches appropriate source storage for the indicator (e.g. storage/memory used for indicators by other commands). The indicator parameter string specifies the indicator string being sought and wildcarding is supported. In one embodiment, appropriate MS storage/memory which contains the history of indicators sent to the source system is searched and all occurrences found in history are presented with at least their date/time stamps, the indicator, and perhaps other information, for user reconciliation at block 6942. In a preferred embodiment, the most recently delivered indicator is identified and used for the copy. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for the copy. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made so that the target system(s) are delivered the indicator(s) like delivering a new indicator for presentation.

Fig. 69B-5

Operand ↓	**PM**	**Preferred embodiment Copy processing**
215	**C**	Copying an application causes searching the source system for the application (and with the params parameter(s) if specified to get the params specified invocation of the application). The app parameter is preferably an executable name and may contain parameters that were passed. Providing a more defined partial or full path to the application parameter will limit the search result. The app parameter string preferably supports wildcarding. In one embodiment, all occurrences found at the source and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation at block 6942. In a preferred embodiment, the most recently executed instance of the matching application is determined for the copy. In one embodiment, the application itself is copied to the target systems, perhaps as directed by an additional parameter (e.g. directory location). In another embodiment, the executable path to run the application is placed into execution history at the system(s) so that a user can run it, albeit from a remote system (assumption that application available for running there already). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copy. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
217	**C**	Copying a document causes searching the source for the document. The doc parameter is a document name. The document parameter can be a wildcard (pattern) for matching. Providing a more defined partial or full path to the document name will narrow the search. In one embodiment, all occurrences found on the MS and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information, for reconciliation at block 6942. In a preferred embodiment, the most recently accessed search result is provided for the copy. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Copying the document places a copy to each target system at a special shared folder, or configured folder for sharing, or as specified with a new destination parameter to copy processing.

Fig. 69B-6

Operand ↓	**PM**	**Preferred embodiment Copy processing**
219	**C**	Copying a file causes searching the source path for the file. The path parameter is a file name. Providing a more defined partial or full path to the file will narrow the search result. The path parameter can be a wildcard (pattern) for matching. In one embodiment, all occurrences found on the MS and their paths are provided with at least their date/time stamps, size, and perhaps attributes information, for reconciliation at block 6942. In a preferred embodiment, the most recently accessed file meeting the search criteria is copied to the target systems. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In one embodiment, the copy results in overwriting an existing file with the same handle (e.g. name). In another embodiment, the copy results in writing a newly altered name of the file when there is an existing file with the same handle (e.g. name). An additional parameter may be specified for the target (specified folder), otherwise the object is copied to an assumed location.
221	**O**	Copying content causes searching the source for the content. The content parameter is a reference to the content. The content parameter can be a wildcard (pattern) for matching. In a preferred embodiment, the most recently accessed search result is provided for the copy. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Copying the content places a copy to each target system at a special shared destination, or configured destination for sharing, or as specified with a new parameter to copy processing.

Fig. 69B-7

Operand ↓	**PM**	**Preferred embodiment Copy processing**
223	**O**	Copying a DB object causes searching the source for the database object value. The database object parameter is provided with a variety of syntaxes depending on the type of database object sought. For example, the DB-obj parameters is "T:tablename" to seek a table, "S:schemaname" to seek a particular schema, "C:columnname" to seek a particular column name, "D:DBname" to seek a particular DB name, "R:rolename" to seek a particular role set, "P:procname" to search for particular stored procedure, etc. There are unique syntaxes for every type of DB object being sought which maps to an appropriate SQL system tables query. The search criteria can be a wildcard (pattern) for matching. In one embodiment, all occurrences found on the source system and information about the occurrence is presented to the user for reconciliation at block 6942. In other embodiments, the best (e.g. most recently accessed) fit database object is identified for use in the copy, or a new parameter indicating how to search. The search takes place as though the user manually launched the search, entered the criteria or query for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The value of the DB object is copied to the value of the DB object with the same name and type at the destination system(s). If not found at a target system, then no action is performed at that system. Copying a database object copies the value to the same database object(s) at other system(s), or creates new copies of the DB objects there when names do not match. Copying a DB object is intended to keep DBs in synch in some uses. Value(s) are overwritten. An additional parameter may be specified for the target of the copy (e.g. schema path which may include credentials for authentication).

Fig. 69B-8

Operand ↓	PM	Preferred embodiment Copy processing
225	O	Copying data causes searching the source system for the data. In the preferred embodiment, the data is a global system variable visible to all processes of a MS operating system. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). Depending on the embodiment, data may be that which is contained in a program data segment, stack segment, and/or extra segment. There can be unique syntaxes for specifying which type of data is being sought (e.g. "S:dataname" for data parameter). The search criteria can be a wildcard (pattern) for matching. In the preferred embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. A recognized AppTerm causes access to record 5300 for proper semaphore synchronized access. In one embodiment, all occurrences found at the source system and information about the occurrence including its current value is presented to the user for reconciliation at block 6942. In a preferred embodiment, the best data value (e.g. most recently accessed if more than one matches) is provided for the copy. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The value of the copied data is copied to the data with the same name and type at the destination system(s). If not found at a target system, then no action is performed at that system, or an error is provided. Copying a data object copies the value to the same data object(s) at other system(s). Copying is intended to keep data in synch between systems in some uses. Value(s) are overwritten. An additional parameter may be specified for the target data name of the copy.
227	O	Copying a semaphore causes reading the current value of the semaphore at the source where the copy command action is being executed and then copying the current value to the same semaphore names at the target system(s). The semaphore param can be a wildcard (pattern) for matching. In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The value (set or cleared) of the copied semaphore is copied to the semaphore with the same name and type at the destination system(s). If not found at a target system, then no action is performed at that system, or an error is provided. Copying a semaphore copies the value to the same semaphore at other system(s). Copying a semaphore is intended to keep systems in synch in some uses. Value(s) are overwritten. An additional parameter may be specified for the target sem name of the copy.

Fig. 69B-9

Operand ↓	PM	Preferred embodiment Copy processing
229	C	Copying a directory causes searching the source system for the directory. The path parameter is a directory name. Providing a more defined partial or full path to the directory parameter will narrow the search result. The path parameter can be a wildcard (pattern) for matching. In one embodiment, all occurrences found on the MS and their paths are provided with at least their date/time stamps, size, and perhaps attributes information, for reconciliation at block 6942. In a preferred embodiment, the most recently accessed directory meeting the search criteria is copied to the target systems. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In one embodiment, the copy results in overwriting an existing directory and files therein. In another embodiment, the copy results in writing a newly altered name of directory contents when there is a conflict (e.g. existing entity with same name). In another embodiment, an additional target path parameter is provided for where to place the directory.
231	C	Operand 215 (application object) is treated identically to this Operand 231 (application context) this LBX release (same params currently).
233	C	Copying a focused user interface object causes capturing the currently focused user interface object using the first parameter (e.g. Alt-Prtscrn; can be changed with the param) string syntax for keystroke(s) to capture the image, and then copying the graphics file (file type in various embodiments) to a shared destination, or a configured destination at the target system(s) or as specified with a new parameter. The capture takes place as though the user manually performed the capture action, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the capture and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In one embodiment, the copy results in overwriting an existing file with the same handle (e.g. name), which will be seldom since the graphics file name preferably contains a date/time stamp portion. In another embodiment, the copy results in writing a newly altered name of the file when there is an existing file with the same handle (e.g. name). An additional parameter may be specified for the target of the copy. An additional parameter may be specified for the target format of the copy whereby a conversion is caused (e.g. JPG to TIFF).
235	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter). Also, the ack parameter provides a reconciliation option.
237	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter). Also, the ack parameter provides a reconciliation option.

Fig. 69B-10

Operand ↓	**PM**	**Preferred embodiment Copy processing**
239	**C**	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter). Also, the ack parameter provides a reconciliation option.
241	**C**	Copying an alert causes searching the source for the alert. The alert parameter is the same parameter used to generate an alert (e.g. using another command), and can be wildcarded. In one embodiment, all occurrences found on the MS which is associated to the alerter application in use at the MS, and which is used for other commands disclosed, are provided to the user for reconciliation at block 6942 with at least their date/time stamps, and perhaps other information. In other embodiments, the most recently generated alert matching the alert search criteria is used for copying, or the search occurs as specified with an additional parameter. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made so that the target system(s) are delivered the alert(s) like delivering a new alert to the systems.
243	**C**	Copying a process causes first finding all process names running at the source (e.g. MS) which contain the prname string parameter (e.g. in UNIX: "ps -ef | grep prname"). In one embodiment, all occurrences found running at the MS are presented with interesting programmatic information such as when started, its size, etc for reconciliation at block 6942. In the preferred embodiment, one process running in the source system is to be found (i.e. >1 = ambiguous). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying. Results are useful statistics about the process which is running at the source. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, historical log, or other useful storage for subsequent use. Useful statistic(s) about the process (perhaps which statistics specified with an additional parameter) are copied to an appropriate destination of the target system(s) for informative purposes (e.g. a special log file). In another embodiment, the alerter process and/or indicator methodology can be used as the destination for the copy for alerting a user at the target system. In another embodiment, there is a new parameter for which end result the copy will have (informative destination, handled like alert, handled like indicator).

Fig. 69B-11

Operand ↓	PM	Preferred embodiment Copy processing
245	C	Copying a container causes searching the source system for the container. The container parameter can be well defined to narrow the search result, and may be wildcarded (pattern) for matching. In one embodiment, all occurrences found on the MS and their references/handles are provided with at least their date/time stamps, size, and perhaps attributes information, for reconciliation at block 6942. In a preferred embodiment, the most recently accessed container meeting the search criteria is copied to the target systems. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In one embodiment, the copy results in overwriting an existing container. In another embodiment, the copy results in writing a newly altered reference/handle of the container when there is a conflict (e.g. existing entity with same name). An additional parameter may be specified for the target, otherwise the object is copied to an assumed location.
247	C	Copying a program object first causes searching the source for the program object. In the preferred embodiment, a unique syntax is used for which type of program object is being sought (similar to above). There can be unique syntaxes for specifying which type of program object is being sought (e.g. "S:dataname"). In one embodiment, all occurrences found on the MS and information about the occurrence including its current value is presented to the user for reconciliation at block 6942. In a preferred embodiment, one program object is to be found (e.g. >1 = ambiguous). In one embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Useful statistic(s) about the program object (perhaps which statistics specified with an additional parameter) are copied to an appropriate destination of the target system(s) for informative purposes (e.g. a special log file). In another embodiment, the alerter process and/or indicator methodology can be used as the destination for the copy for alerting a user at the target system. In another embodiment, there is new parameter for which end result the copy will have (informative destination, handled like alert, handled like indicator). An alternate embodiment works like Operand 223 wherein copying is intended to keep program object(s) between systems in synch. Such embodiments require source and target system processing to have access to the object(s) (this may limit participating object(s)).

Fig. 69B-12

Operand ↓	PM	Preferred embodiment Copy processing
249	**C**	Copying a cursor causes searching the source system for the current cursor setting(s). In the preferred embodiment, provided in the search results is the current cursor information (e.g. image, animation , etc). The current cursor information of the source is then used to alter the cursor at the target system(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the copy operation. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
251	**C**	Copying a calendar (CAL) object causes searching the source CAL system with search criteria of the calendar object parameter string. The calendar object parameter string can specify searching any calendar entry fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria (similar to email above). Those skilled in the art recognize many useful syntaxes for searching any characteristics of CAL objects. In one embodiment, all occurrences found in history are presented to the user for reconciliation at block 6942 with at least their date/time stamps, attendees, and perhaps other information, of the CAL object and when it was scheduled. In a preferred embodiment, the most recent occurrence from the calendaring system is provided for the copy. Wildcarding (pattern matching) is preferably inherent by searching for substrings. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The calendar entry is copied in its entirety to the target system calendaring system. In another embodiment, a new parameter is specified to copy the calendar item to a new schedule or time. A duplicate calendar entry may be created if one already exists.

Fig. 69B-13

Operand ↓	**PM**	**Preferred embodiment Copy processing**
253	**C**	Copying an address book (AB) object causes searching the source AB system with search criteria of the AB object parameter string. The AB object parameter string can specify searching any AB entry fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria (similar to email above). Those skilled in the art recognize many useful syntaxes for searching any characteristics of AB objects/entries. In one embodiment, all occurrences found are presented to the user for reconciliation at block 6942 with appropriate AB information. In a preferred embodiment, the most recent occurrence from the AB system is provided for the copy. Wildcarding (pattern matching) is preferably inherent by searching for substrings. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The AB entry is copied in its entirety to the target system AB system. In another embodiment, a new parameter is specified to copy the AB item to special destination. A duplicate AB entry may be created if one already exists.
. . .		

Fig. 69B-14

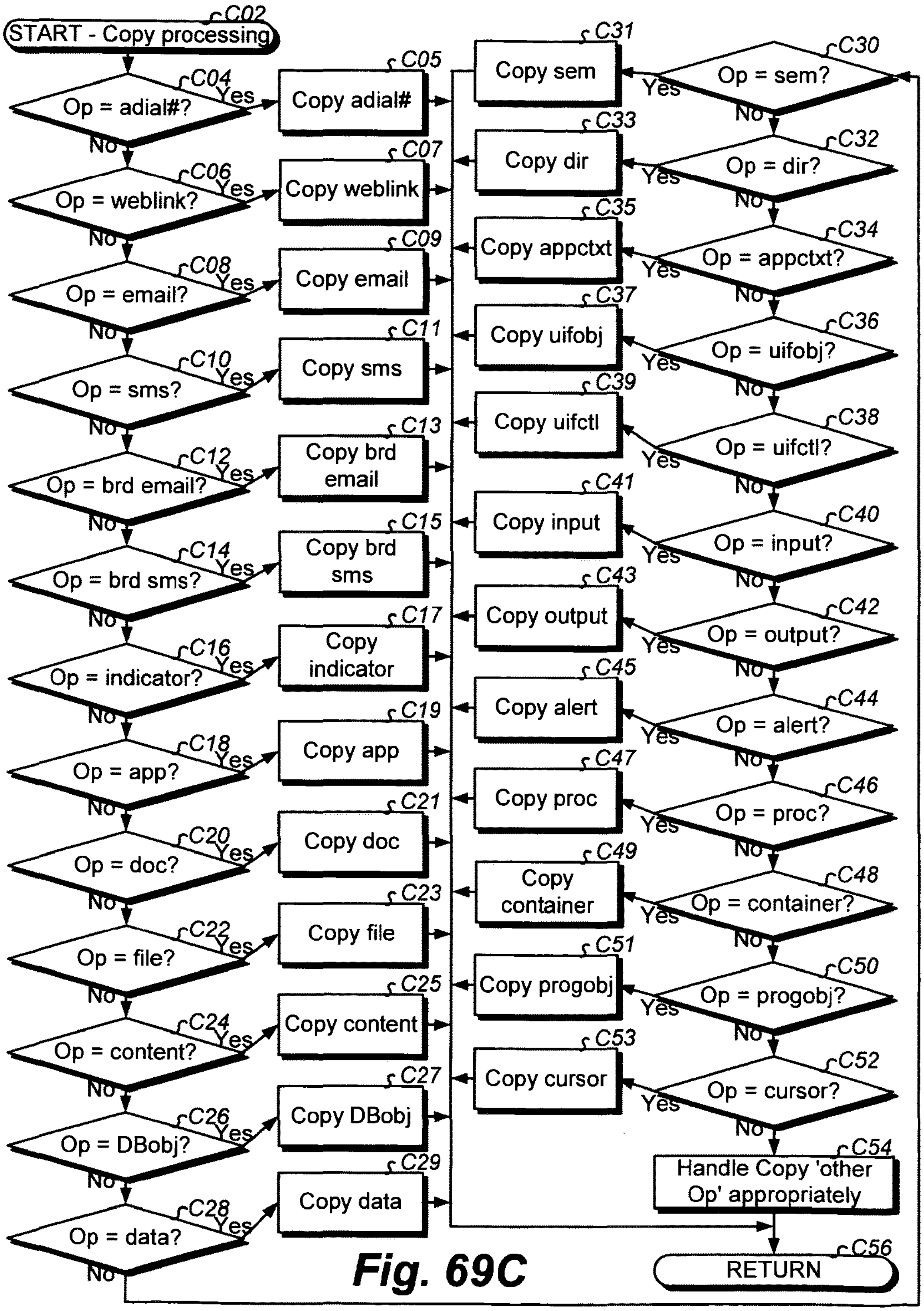

C02
START - Copy processing
C04
Op = adial#?
Yes
No
C05
Copy adial#
C06
Op = weblink?
C07
Copy weblink
C08
Op = email?
C09
Copy email
C10
Op = sms?
C11
Copy sms
C12
Op = brd email?
C13
Copy brd email
C14
Op = brd sms?
C15
Copy brd sms
C16
Op = indicator?
C17
Copy indicator
C18
Op = app?
C19
Copy app
C20
Op = doc?
C21
Copy doc
C22
Op = file?
C23
Copy file
C24
Op = content?
C25
Copy content
C26
Op = DBobj?
C27
Copy DBobj
C28
Op = data?
C29
Copy data
C30
Op = sem?
C31
Copy sem
C32
Op = dir?
C33
Copy dir
C34
Op = appctxt?
C35
Copy appctxt
C36
Op = uifobj?
C37
Copy uifobj
C38
Op = uifctl?
C39
Copy uifctl
C40
Op = input?
C41
Copy input
C42
Op = output?
C43
Copy output
C44
Op = alert?
C45
Copy alert
C46
Op = proc?
C47
Copy proc
C48
Op = container?
C49
Copy container
C50
Op = progobj?
C51
Copy progobj
C52
Op = cursor?
C53
Copy cursor
C54
Handle Copy 'other Op' appropriately
C56
RETURN

Fig. 69C

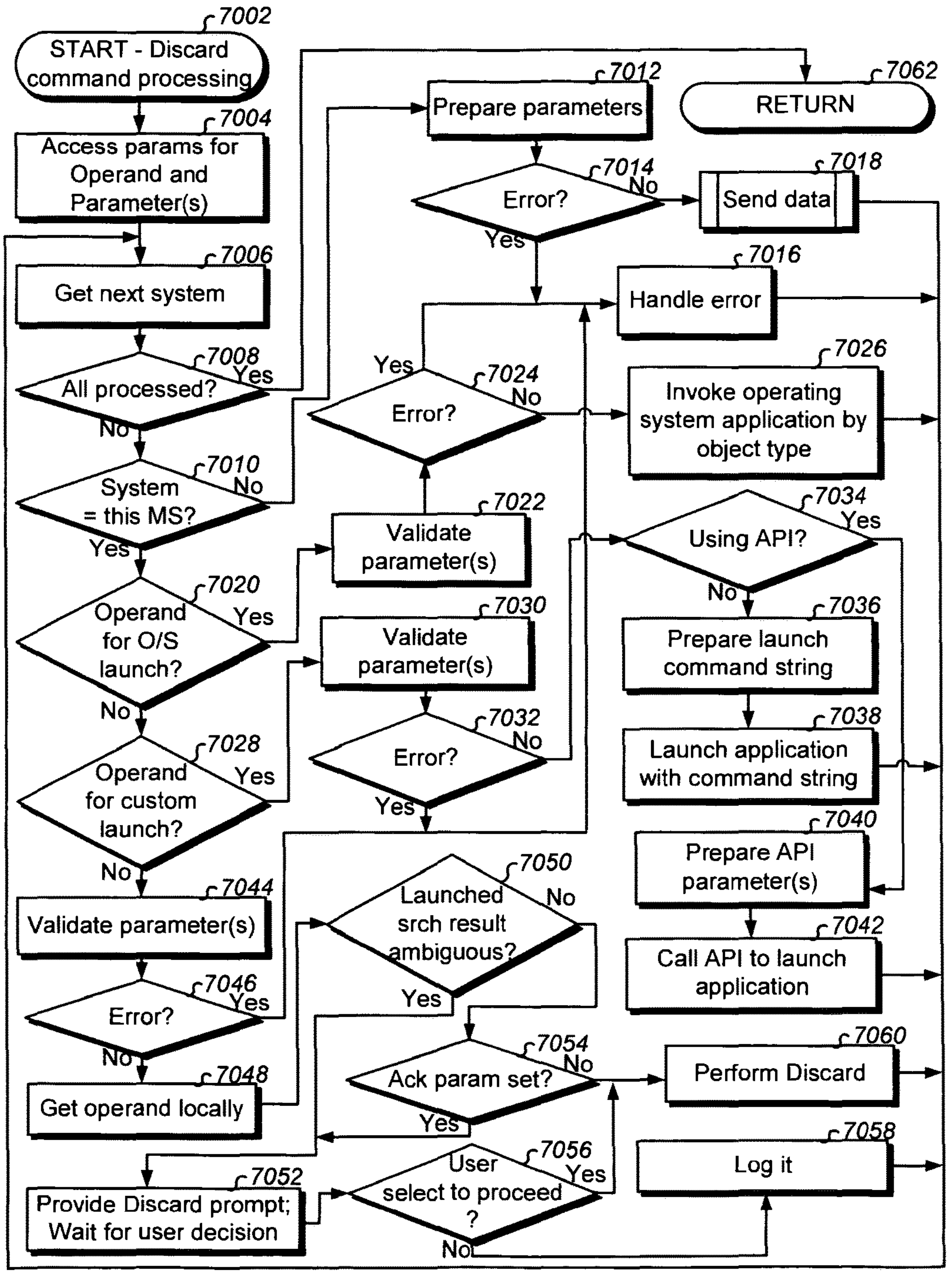

7002
START - Discard command processing
7004
Access params for Operand and Parameter(s)
7006
Get next system
7008
All processed?
Yes
No
7010
System = this MS?
No
Yes
7020
Operand for O/S launch?
Yes
No
7028
Operand for custom launch?
Yes
No
7044
Validate parameter(s)
7046
Error?
Yes
No
7048
Get operand locally
7052
Provide Discard prompt; Wait for user decision
7012
Prepare parameters
7014
Error?
No
Yes
7018
Send data
7062
RETURN
7016
Handle error
7024
Error?
Yes
No
7026
Invoke operating system application by object type
7022
Validate parameter(s)
7030
Validate parameter(s)
7032
Error?
No
Yes
7034
Using API?
Yes
No
7036
Prepare launch command string
7038
Launch application with command string
7040
Prepare API parameter(s)
7042
Call API to launch application
7050
Launched srch result ambiguous?
No
Yes
7054
Ack param set?
No
Yes
7060
Perform Discard
7056
User select to proceed ?
Yes
No
7058
Log it

Fig. 70A

Operand ↓	**PM**	**Preferred embodiment Discard processing**
201	**C**	Discarding an auto-dial # launches a phone number log interface with the auto-dial # parameter (can be wildcarded) for searching the specified system (e.g. MS). Preferably, both the outgoing and incoming logs are searched. In an alternate embodiment, the log is specified with a parameter. In the preferred embodiment, the most recent occurrence from a particular log is to be discarded. In another embodiment, all occurrences found in history are presented with their date/time stamps, and perhaps other information, of the call and when it took place when the ack parameter is set and the user browses the results prior to accepting the discard of multiple items. The search takes place as though the user manually launched the search, entered the auto-dial # for the search, and then was provided with result(s) for the discard. Preferably, the discard shall take place if there are no ambiguities (e.g. more than one phone number returned per search criteria). Appropriate MS storage is updated and subsequently processed as though the user manually performed the search and discard of the result. Preferably, the discard cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The discard is made to system(s) logs.
203	**S**	Discarding a weblink launches a search to browser history with the weblink parameter (can be wildcarded) for searching the specified system. In one embodiment, all occurrences found in history are presented with their date/time stamps, and perhaps other information, of the link and when it was invoked, when the ack parameter is specified to true for presentation to the user prior to doing the discard. In the preferred embodiment, the most recent occurrence from a particular invocation is provided for the discard. The search takes place as though the user manually launched the search, entered the weblink for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard of the result. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The discard is made to a special browser favorites folder, another designated folder configured ahead of time, or as specified with an additional parameter.

Fig. 70B-1

Operand ↓	**PM**	**Preferred embodiment Discard processing**
205	**C**	Discarding an email causes searching the specified email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding (patterns for matching). Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel';recip:'george@alltell.com';body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place, when the ack parameter is set to true for user reconciliation. In another embodiment, the most recent occurrence from searched folders is provided for the discard. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for doing the discard. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The email is discarded from an email folder as specified with a syntax in the email parameter string.

Fig. 70B-2

Operand ↓	PM	Preferred embodiment Discard processing
207	C	Discarding an sms message causes searching the specified messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place, when the ack parameter is set to true for user reconciliation. In another embodiment, the most recent occurrence from searched folders is provided for discarding. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The message is discarded from a folder as specified with a syntax in the sms message parameter string.

Fig. 70B-3

Operand ↓	PM	Preferred embodiment Discard processing
209	C	Discarding a broadcast email causes searching the specified email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding (patterns for matching). Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel';recip:'george@alltell.com';body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for the discard. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for doing the discard. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The email is discarded from a folder as specified with a syntax in the email parameter string.

Fig. 70B-4

Operand ↓	**PM**	**Preferred embodiment Discard processing**
211	**C**	Discarding a broadcast sms msg causes searching the specified messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for discarding. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The message is discarded from a folder as specified with a syntax in the sms message parameter string.
213	**O**	Discarding an indicator searches appropriate specified system storage for the indicator (e.g. storage/memory used for indicators by other commands). The indicator parameter string specifies the indicator string being sought and wildcarding is supported. In one embodiment, appropriate MS storage/memory which contains the history of indicators sent to the source system is searched and all occurrences found in history are presented with at least their date/time stamps, the indicator, and perhaps other information, for user reconciliation. In one embodiment, the most recently delivered indicator is identified and used for the discard. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for the discard. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The discard is performed so that the target system(s) have the indicator(s) removed from the interface if currently presented and removed from history maintained for the user interface object presentation (preferably not from LBX history). An additional parameter may specify how to delete the indicator.

Fig. 70B-5

Operand ↓	**PM**	**Preferred embodiment Discard processing**
215	**C**	Discarding an application causes searching the specified system for the application (and with the params parameter(s) if specified to get the right invocation of the application). The app parameter is preferably an executable name. Providing a partial or full path to the application parameter will limit the search result. The app parameter string preferably supports wildcarding. In one embodiment, all occurrences found at the source and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In a preferred embodiment, the most recently executed instance of the matching application is determined for the discard. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provide with the result for discard. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The discard does not remove the application from the target system. It terminates the application by terminating/killing it at the operating system level. A Discard File operand command can be used to remove it from the system.
217	**S**	Discarding a document causes searching the specified system for the document. The doc parameter is a document name. The document parameter can be a wildcard (pattern) for matching. Providing a more defined partial or full path to the document name will narrow the search. In one embodiment, all occurrences found and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In a preferred embodiment, the most recently accessed search result is provided for the discard. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Discard of the document removes it from each target system at a special shared folder, or configured folder for sharing, or as specified with a new parameter to discard processing.
219	**S**	Discarding a file causes searching the source path for the file. The path parameter is a file name. Providing a more defined partial or full path to the file will narrow the search result. The path parameter can be a wildcard (pattern) for matching. In one embodiment, all occurrences found on the MS and their paths are provided with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In another embodiment, the most recently accessed file meeting the search criteria is discarded. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provide with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. File(s) are discarded at each target system.

Fig. 70B-6

Operand ↓	**PM**	**Preferred embodiment Discard processing**
221	**O**	Discarding content causes searching the specified system for the content. The content parameter is a reference to the content. The content parameter can be a wildcard (pattern) for matching. In a preferred embodiment, the most recently accessed search result is provided for the discard. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Discarding the content removes it from each target system at a special shared destination, or configured destination for sharing, or as specified with a new parameter to discard processing.
223	**C**	Discarding a DB object causes searching the specified system for the database object value. The database object parameter is provided with a variety of syntaxes depending on the type of database object sought. For example, the DB-obj parameters is "T:tablename" to seek a table, "S:schemaname" to seek a particular schema, "C:columnname" to seek a particular column name, "D:DBname" to seek a particular DB name, "R:rolename" to seek a particular role set, "P:procname" to search for particular stored procedure, etc. There are unique syntaxes for every type of DB object being sought. The search criteria can be a wildcard (pattern) for matching. In one embodiment, all occurrences found on the source system and information about the occurrence is presented to the user for reconciliation. In a preferred embodiment, the best (e.g. most recently accessed) fit database object is identified for discard. The search takes place as though the user manually launched the search, entered the criteria or query for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The DB object is discarded (removed, deleted, dropped).
225	**O**	Same as Compose processing except modifies the value to an initial value (e.g. 0) at each system.
227	**O**	Same as Compose processing except modifies the value to an initial value (e.g. clear) at each system.

Fig. 70B-7

Operand ↓	PM	Preferred embodiment Discard processing
229	S	Discarding a directory causes searching the specified system for the directory. The path parameter is a directory name. Providing a more defined partial or full path to the directory parameter will narrow the search result. The path parameter can be a wildcard (pattern) for matching. In one embodiment, all occurrences found on the MS and their paths are provided with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In one embodiment, the most recently accessed directory meeting the search criteria is discarded. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The directory is discarded at each target system.
231	C	Operand 215 and 235 (application object) is treated identically to Operand 231 (application context) this LBX release (same params currently). The specified application is terminated, not removed.
233	S	Discarding a user interface object causes closing/terminating the focused object(s) at each specified system that contains the objtxt parameter criteria in the titlebar. In a preferred embodiment, there is a unique syntax for which places of user interface objects that are currently active are to be search (e.g. title bar, entry fields, radio button options, window text, combinations thereof, etc). The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched the search, entered the criteria for the search, and the object(s) are closed/terminated. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
235	C	Operand 215 and 231 (application object) is treated identically to Operand 235 (application context) this LBX release (same params currently). The specified application is terminated, not removed.
237	O	Discarding input causes reinitializing the iodev parameter input device stream of the specified system, so that any pending state is discarded. In one embodiment, a special input datastream is issued to reinitialize the I/O path. In another embodiment, the I/O path is terminated and restarted to reinitialize for the attached device(s). In another embodiment, the I/O path is flushed and then reinitialized. In another embodiment, an additional parameter indicates how to discard the iodev device stream (e.g. method and/or initialization data to initialize with). The iodev parameter specifies which Input/Output device to reinitialize. Preferably, the discard command data is maintained to LBX History, a log, or other useful storage for subsequent use.

Fig. 70B-8

Operand ↓	PM	Preferred embodiment Discard processing
239	O	Discarding output causes reinitializing the iodev parameter output device stream of the specified system, so that any pending state is discarded. In one embodiment, a special output datastream is issued to reinitialize the I/O path. In another embodiment, the I/O path is terminated and restarted to reinitialize for the attached device(s). In another embodiment, the I/O path is flushed and then reinitialized. In another embodiment, an additional parameter indicates how to discard the iodev device stream (e.g. method and/or initialization data to initialize with). The iodev parameter specifies which Input/Output device to reinitialize. Preferably, the discard command data is maintained to LBX History, a log, or other useful storage for subsequent use.
241	C	Discarding an alert causes searching the specified system for the alert and discarding it. The alert parameter is the same parameter used to generate an alert (e.g. using another command), and can be wildcarded. In one embodiment, all occurrences found which is associated to the alerter application in use at the MS, and which is used for other commands disclosed, are provided to the user for reconciliation with at least their date/time stamps, and perhaps other information. In one embodiment, the most recently generated alert matching the alert search criteria is used for discarding. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
243	O	Discarding a process causes searching for and terminating/killing all process names running at the specified system (e.g. MS) which contain the prname string parameter (e.g. in UNIX: "ps -ef | grep prname"). In one embodiment, all occurrences found running at the MS are presented with interesting programmatic information such as when started, its size, etc for user reconciliation. In the preferred embodiment, one process running in the specified system is to be found (i.e. >1 = ambiguous). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, historical log, or other useful storage for subsequent use. The process is terminated/killed. It is not removed from the system.

Fig. 70B-9

Operand ↓	PM	Preferred embodiment Discard processing
245	S	Discarding a container causes searching the specified system for the container. The container parameter can be well defined to narrow the search result, and may be wildcarded (pattern) for matching. In one embodiment, all occurrences found on the MS and their references/handles are provided with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In one embodiment, the most recently accessed container meeting the search criteria is discarded at the target systems. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The container is discarded from each specified system.
247	O	Discarding a program object causes searching the specified system for the program object and initializing to a initial value (i.e. discarding any current value). In the preferred embodiment, a unique syntax is used for which type of program object is being sought (similar to above). There can be unique syntaxes for specifying which type of program object is being sought (e.g. "S:dataname"). In one embodiment, all occurrences found on the MS and information about the occurrence including its current value is presented to the user for reconciliation. In a preferred embodiment, one program object is to be found (e.g. >1 = ambiguous). In one embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. A reasonable reset value (e.g. 0 for data, clear for semaphore) is set to the program object. Program objects which cannot hold a value (procedure) are preferably not affected by the discard command. Local and remote processing must have programmatic visibility to affected program object(s).
249	O	Discarding a cursor causes resetting the cursor at the specified system. In the preferred embodiment, provided in the search results is the current cursor information (e.g. image, animation , etc) when user reconciliation is involved. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the discard operation. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In one embodiment, a system defaulted cursor is set. In another embodiment, the previous cursor setting is returned to, and multiple discard cursor actions can change the cursor continuously to historical settings. An additional parameter may provide how to discard the cursor.

Fig. 70B-10

Operand ↓	PM	Preferred embodiment Discard processing
251	C	Discarding a calendar object causes searching the specified calendar system with search criteria of the calendar object parameter string. The calendar object parameter string can specify searching any calendar entry fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria (similar to email above). Those skilled in the art recognize many useful syntaxes for searching any characteristics of calendar objects. In one embodiment, all occurrences found in history are presented to the user for reconciliation with at least their date/time stamps, sender and recipient, and perhaps other information, of the calendar object and when it was scheduled. In a preferred embodiment, the most recent occurrence from the calendaring system is discarded. Wildcarding (pattern matching) is preferably inherent by searching for substrings. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The calendar entry(s) are discarded from the target system calendaring system.
253	C	Discarding an address book (AB) object causes searching the specified AB system with search criteria of the AB object parameter string. The AB object parameter string can specify searching any AB entry fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria (similar to email above). Those skilled in the art recognize many useful syntaxes for searching any characteristics of AB objects/entries. In one embodiment, all occurrences found are presented to the user for reconciliation with appropriate AB information. In a preferred embodiment, the most recent occurrence from the AB system is discarded. Wildcarding (pattern matching) is preferably inherent by searching for substrings. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The AB entry(s) are discarded from the target system AB system.
. . .		

Fig. 70B-11

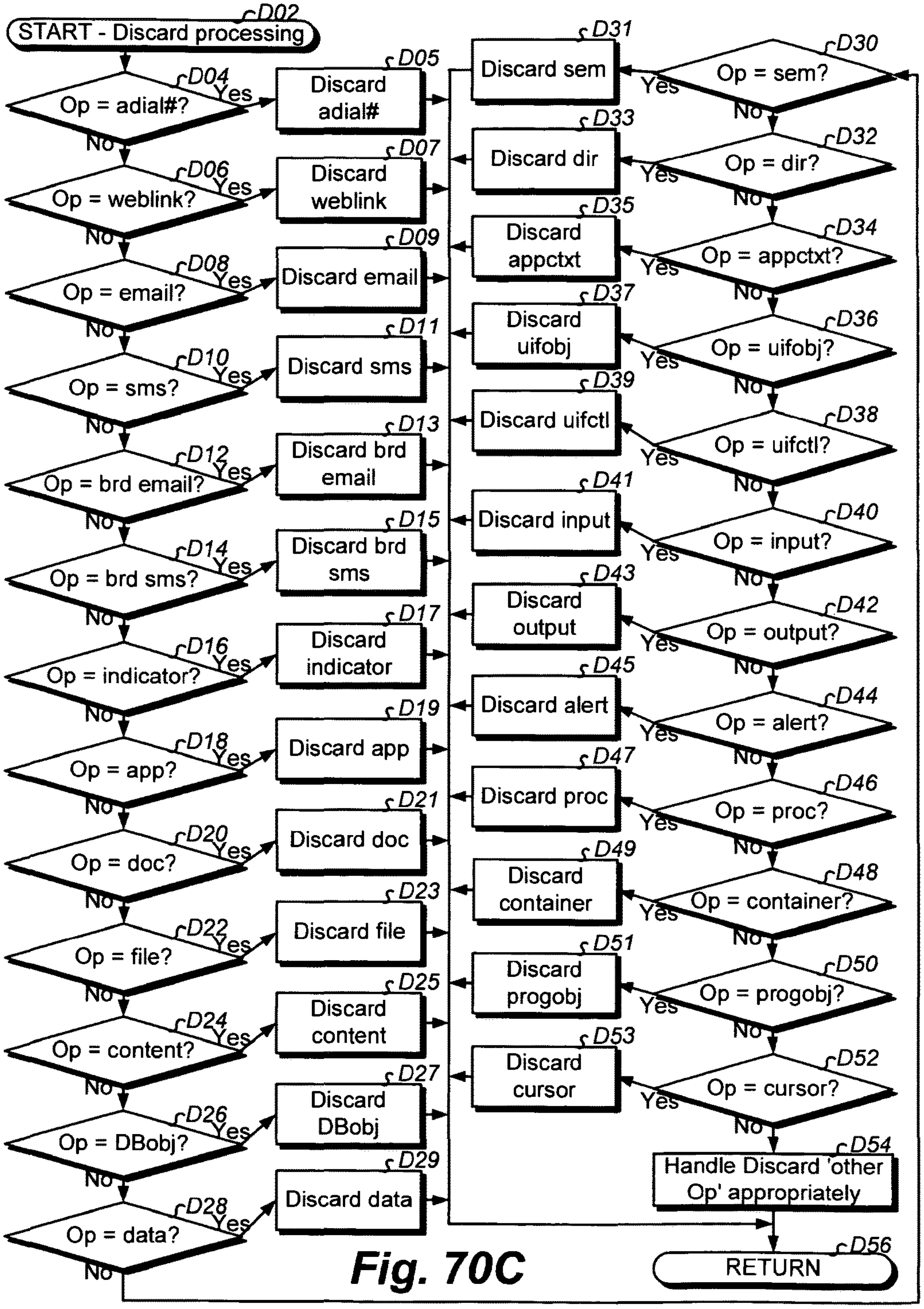

START - Discard processing
D02
Op = adial#?
D04
Yes
No
Discard adial#
D05
Op = weblink?
D06
Discard weblink
D07
Op = email?
D08
Discard email
D09
Op = sms?
D10
Discard sms
D11
Op = brd email?
D12
Discard brd email
D13
Op = brd sms?
D14
Discard brd sms
D15
Op = indicator?
D16
Discard indicator
D17
Op = app?
D18
Discard app
D19
Op = doc?
D20
Discard doc
D21
Op = file?
D22
Discard file
D23
Op = content?
D24
Discard content
D25
Op = DBobj?
D26
Discard DBobj
D27
Op = data?
D28
Discard data
D29
Op = sem?
D30
Discard sem
D31
Op = dir?
D32
Discard dir
D33
Op = appctxt?
D34
Discard appctxt
D35
Op = uifobj?
D36
Discard uifobj
D37
Op = uifctl?
D38
Discard uifctl
D39
Op = input?
D40
Discard input
D41
Op = output?
D42
Discard output
D43
Op = alert?
D44
Discard alert
D45
Op = proc?
D46
Discard proc
D47
Op = container?
D48
Discard container
D49
Op = progobj?
D50
Discard progobj
D51
Op = cursor?
D52
Discard cursor
D53
Handle Discard 'other Op' appropriately
D54
RETURN
D56

Fig. 70C

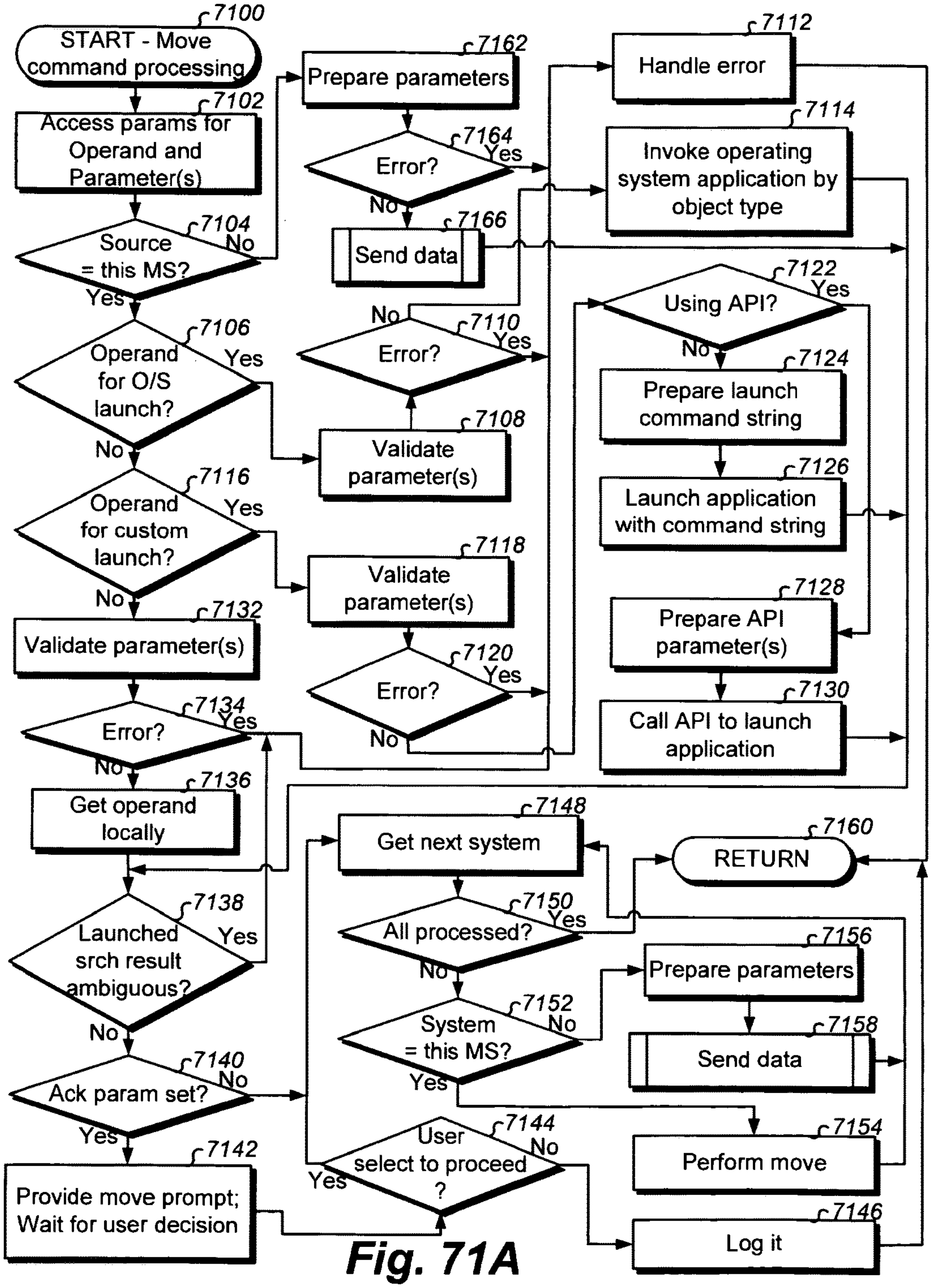

7100
START - Move command processing
7102
Access params for Operand and Parameter(s)
7104
Source = this MS?
No
Yes
7106
Operand for O/S launch?
Yes
No
7116
Operand for custom launch?
Yes
No
7132
Validate parameter(s)
7134
Error?
Yes
No
7136
Get operand locally
7138
Launched srch result ambiguous?
Yes
No
7140
Ack param set?
No
Yes
7142
Provide move prompt; Wait for user decision
7162
Prepare parameters
7164
Error?
Yes
No
7166
Send data
No
7110
Error?
Yes
7108
Validate parameter(s)
7118
Validate parameter(s)
7120
Error?
Yes
No
7112
Handle error
7114
Invoke operating system application by object type
7122
Using API?
Yes
No
7124
Prepare launch command string
7126
Launch application with command string
7128
Prepare API parameter(s)
7130
Call API to launch application
7148
Get next system
7160
RETURN
7150
All processed?
Yes
No
7156
Prepare parameters
7152
System = this MS?
No
Yes
7158
Send data
7144
User select to proceed ?
No
Yes
7154
Perform move
7146
Log it

Fig. 71A

Operand ↓	PM	Preferred embodiment Move processing
201	C	Moving an auto-dial # launches a phone number log interface with the auto-dial # parameter (can be wildcarded) for searching the source system. Preferably, both the outgoing and incoming logs are searched. In an alternate embodiment, the log is specified with a parameter. In the preferred embodiment, the most recent occurrence from a particular log is provided. In another embodiment, all occurrences found in history are presented with their date/time stamps, and perhaps other information, of the call and when it took place (e.g. when the ack parameter is set) and the user browses the results prior to accepting the move. The search takes place as though the user manually launched the search, entered the auto-dial # for the search, and then was provided with result(s) for the move. Preferably, the move shall take place if there are no ambiguities (e.g. more than one phone number returned per search criteria). An additional parameter may be specified for the target (different log) of the move, otherwise the object is moved to an assumed location (e.g. same folder to more recent position). Appropriate MS storage is updated and subsequently processed as though the user manually performed the search and move of the result. Preferably, the move cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The move is made to system(s) logs, preferably with identifying information of the source and who did the move.
203	C	Moving a weblink launches a search to MS browser history with the weblink parameter (can be wildcarded) for searching the source system. In one embodiment, all occurrences found in history are presented with their date/time stamps, and perhaps other information, of the link and when it was invoked (e.g. when the ack parameter is specified to true) for presentation to the user prior to doing the move. In the preferred embodiment, the most recent occurrence from a particular invocation is provided for the move. An additional parameter may be specified for the target (specified favorites folder), otherwise the object is moved to an assumed location (e.g. highest level favorites folder). The search takes place as though the user manually launched the search, entered the weblink for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move of the result. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The move is made to a special browser favorites folder, or another designated folder configured ahead of time, or as specified with an additional parameter, preferably with identifying information of the source and who did the move.

Fig. 71B-1

Operand ↓	PM	Preferred embodiment Move processing
205	C	Moving an email causes searching the source email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding (patterns for matching). Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel';recip:'george@alltell.com';body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for the move. An additional parameter may be specified for the target (specified folder), otherwise the object is moved to an assumed location (e.g. inbox, drafts, special named folder).The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for doing the move. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The move is made to a special email folder of the target system, or another designated folder configured ahead of time, or as specified with a new parameter for move processing, preferably with identifying information of the source and who did the move (if supported in email application).

Fig. 71B-2

Operand ↓	PM	Preferred embodiment Move processing
207	**C**	Moving an sms message causes searching the source messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for moving. An additional parameter may be specified for the target (specified folder), otherwise the object is moved to an assumed location (e.g. inbox, drafts, special named folder). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The move is made to a special messaging folder of the target system, or another designated folder configured ahead of time, or as specified with a new parameter to move processing, preferably with identifying information of the source and who did the move.

Fig. 71B-3

Operand ↓	**PM**	**Preferred embodiment Move processing**
209	**C**	Moving a broadcast email causes searching the source email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding (patterns for matching). Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel';recip:'george@alltell.com';body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for the move. An additional parameter may be specified for the target (specified folder), otherwise the object is moved to an assumed location (e.g. inbox, drafts, special named folder). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for doing the move. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The move is made to a special email folder of the target system, or another designated folder configured ahead of time, or as specified with a new parameter to move processing, preferably with identifying information of the source and who did the move.

Fig. 71B-4

Operand ↓	PM	Preferred embodiment Move processing
211	C	Moving a broadcast sms msg causes searching the source messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for moving. An additional parameter may be specified for the target (specified folder), otherwise the object is moved to an assumed location (e.g. inbox, drafts, special named folder). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The move is made to a special messaging folder of the target system, or another designated folder configured ahead of time, or as specified with a new parameter to move processing, preferably with identifying information of the source and who did the move.
213	C	Moving an indicator searches appropriate source storage for the indicator (e.g. storage/memory used for indicators by other commands). The indicator parameter string specifies the indicator string being sought and wildcarding is supported. In one embodiment, appropriate MS storage/memory which contains the history of indicators sent to the source system is searched and all occurrences found in history are presented with at least their date/time stamps, the indicator, and perhaps other information, for user reconciliation. In a preferred embodiment, the most recently delivered indicator is identified and used for the move. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for the move. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The move is made so that the target system(s) are delivered the indicators like delivering new indicator(s) for presentation.

Fig. 71B-5

Operand ↓	**PM**	**Preferred embodiment Move processing**
215	**C**	Moving an application causes searching the source system for the application (and with the params parameter(s) if specified to get the param specified invocation of the application). The app parameter is preferably an executable name, and may contain parameters that were passed. Providing a more defined partial or full path to the application parameter will limit the search result. The app parameter string preferably supports wildcarding. In one embodiment, all occurrences found at the source and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In a preferred embodiment, the most recently executed instance of the matching application is determined for the move. In one embodiment, the application itself is moved to the target systems, perhaps as directed by an additional parameter (e.g. directory location). In another embodiment, the executable path to run the application is moved to execution history at the system(s) so that a user can run it, albeit from a remote system (assumption that application available for running there already). In another embodiment, the executable(s) are roved to the target system using methodologies of U.S. Patent 5,938,722 ("Method of executing programs in a network", Johnson). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for move. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
217	**C**	Moving a document causes searching the source system for the document. The doc parameter is a document name. The document parameter can be a wildcard (pattern) for matching. Providing a more defined partial or full path to the document name will narrow the search. In one embodiment, all occurrences found on the MS and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In a preferred embodiment, the most recently accessed search result is provided for the move. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Moving the document places the document to each target system at a special shared folder, or configured folder for sharing, or as specified with a new destination parameter to move processing.

Fig. 71B-6

Operand ↓	PM	Preferred embodiment Move processing
219	C	Moving a file causes searching the source path for the file. The path parameter is a file name. Providing a more defined partial or full path to the file will narrow the search result. The path parameter can be a wildcard (pattern) for matching. In one embodiment, all occurrences found on the MS and their paths are provided with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In a preferred embodiment, the most recently accessed file meeting the search criteria is moved to the target systems. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In one embodiment, the move results in overwriting an existing file with the same handle (e.g. name). In another embodiment, the move may result in writing a newly altered name of the file when there is an existing file with the same handle (e.g. name). An additional parameter may be specified for the target (specified folder), otherwise the object is moved to an assumed location.
221	O	Moving content causes searching the source for the content. The content parameter is a reference to the content. The content parameter can be a wildcard (pattern) for matching. In a preferred embodiment, the most recently accessed search result is provided for the move. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Moving the content places the content to each target system at a special shared destination, or configured destination for sharing, or as specified with a new parameter to move processing.

Fig. 71B-7

Operand ↓	PM	Preferred embodiment Move processing
223	O	Moving a DB object causes searching the source for the database object value. The database object parameter is provided with a variety of syntaxes depending on the type of database object sought. For example, the DB-obj parameters is "T:tablename" to seek a table, "S:schemaname" to seek a particular schema, "C:columnname" to seek a particular column name, "D:DBname" to seek a particular DB name, "R:rolename" to seek a particular role set, "P:procname" to search for particular stored procedure, etc. There are unique syntaxes for every type of DB object being sought which maps to an appropriate SQL system tables query. The search criteria can be a wildcard (pattern) for matching. In one embodiment, all occurrences found on the source system and information about the occurrence is presented to the user for reconciliation. In other embodiments, the best (e.g. most recently accessed) fit database object is identified for use in the move, or a new parameter indicates how to search. The search takes place as though the user manually launched the search, entered the criteria or query for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The value of the DB object is moved to the value of the DB object with the same name and type at the destination system(s). If not found at a target system, then no action is performed at that system. Moving a database object moves the value to the same database object(s) at other system(s), or creates new ones when there is not match. Value(s) are overwritten. An additional parameter may be specified for the target of the move.

Fig. 71B-8

Operand ↓	**PM**	**Preferred embodiment Move processing**
225	**O**	Moving data causes searching the source system for the data. In the preferred embodiment, the data is a global system variable visible to all processes of a MS operating system. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). Depending on the embodiment, data may be that which is contained in a program data segment, stack segment, and/or extra segment. There can be unique syntaxes for specifying which type of data is being sought (e.g. "S:dataname" for data parameter). The search criteria can be a wildcard (pattern) for matching. In the preferred embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. A recognized AppTerm causes access to record 5300 for proper semaphore synchronized access. In one embodiment, all occurrences found at the source system and information about the occurrence including its current value is presented to the user for reconciliation. In a preferred embodiment, the best data value (e.g. most recently accessed if more than one matches) is provided for the move. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The value of the data to be moved is copied to the data with the same name and type at the destination system(s). In another embodiment, the source data is reinitialized (e.g. 0), or reinitialized according to a new parameter, as part of the move operation. If not found at a target system, then no action is performed at that system, or an error is provided. Moving a data object at least copies the value to the same data object(s) at other system(s). Value(s) are overwritten. An additional parameter may be specified for the target of the move.

Fig. 71B-9

Operand ↓	PM	Preferred embodiment Move processing
227	O	Moving a semaphore causes reading the current value of the semaphore at the source where the move command action is being executed and then moving the current value to the same semaphore names at the target system(s). The semaphore param can be a wildcard (pattern) for matching. In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The value (set or cleared) of the moved semaphore is copied to the semaphore with the same name and type at the destination system(s). In another embodiment, the source semaphore is reinitialized (e.g. clear), or reinitialized according to a new parameter, as part of the move operation. If not found at a target system, then no action is performed at that system, or an error is provided. Moving a semaphore at least copies the value to the same semaphore at other system(s). Value(s) are changed (clear or set). An additional parameter may be specified for the target of the move.
229	C	Moving a directory causes searching the source system for the directory. The path parameter is a directory name. Providing a more defined partial or full path to the directory parameter will narrow the search result. The path parameter can be a wildcard (pattern) for matching. In one embodiment, all occurrences found on the MS and their paths are provided with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In a preferred embodiment, the most recently accessed directory meeting the search criteria is copied to the target systems. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In one embodiment, the move results in overwriting an existing directory and files therein. In another embodiment, the move results in writing a newly altered name of directory contents when there is a conflict (e.g. existing entity with same name). In another embodiment, an additional target path parameter is provided for where to place the directory.
231	C	Operand 215 (application object) is treated identically to this Operand 231 (application context) this LBX release (same params currently).

Fig. 71B-10

Operand ↓	PM	Preferred embodiment Move processing
233	C	Moving a focused user interface object causes capturing the currently focused user interface object using the first parameter (e.g. Alt-Prtscrn; can be changed with the param) string syntax for keystroke(s) to capture the image, and then moving the graphics file (file type in various embodiments) to a shared destination, or a configured destination at the target system(s), or as specified with a new parameter. The capture takes place as though the user manually performed the capture action, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the capture and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In one embodiment, the move results in overwriting an existing file with the same handle (e.g. name), which will be seldom since the graphics file name preferably contains a date/time stamp portion. In another embodiment, the move results in writing a newly altered name of the file when there is an existing file with the same handle (e.g. name). An additional parameter may be specified for the target of the move. An additional parameter may be specified for the target format of the move whereby a conversion is caused (e.g. JPG to TIFF).
235	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter). Also, the ack parameter provides a reconciliation option.
237	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter). Also, the ack parameter provides a reconciliation option.
239	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter). Also, the ack parameter provides a reconciliation option.
241	C	Moving an alert causes searching the source system for the alert. The alert parameter is the same parameter used to generate an alert (e.g. using another command), and can be wildcarded. In one embodiment, all occurrences found on the MS which is associated to the alerter application in use at the MS, and which is used for other commands disclosed, are provided to the user for reconciliation with at least their date/time stamps, and perhaps other information. In other embodiments, the most recently generated alert matching the alert search criteria is used for moving, or the search occurs as specified with an additional parameter. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The move is made so that the target system(s) are delivered the alert(s) like delivering a new alert to the systems.

Fig. 71B-11

Operand ↓	PM	Preferred embodiment Move processing
243	C	Moving a process causes first finding all process names running at the source (e.g. MS) which contain the prname string parameter (e.g. in UNIX: "ps -ef | grep prname"). In one embodiment, all occurrences found running at the MS are presented with interesting programmatic information such as when started, its size, etc for user reconciliation. In the preferred embodiment, one process running in the source system is to be found (i.e. >1 = ambiguous). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving. Results are useful statistics about the process which is running at the source. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, historical log, or other useful storage for subsequent use. Useful statistic(s) about the process (perhaps which statistics specified with an additional parameter) are moved/copied to an appropriate destination of the target system(s) for informative purposes (e.g. a special log file). In another embodiment, the alerter process and/or indicator methodology can be used as the destination for the move for alerting a user at the target system. In another embodiment, there is a new parameter for which end result the move will have (informative destination, handled like alert, handled like indicator). In some embodiments, the process is roved to the target system using methodologies of U.S. Patent 5,938,722 ("Method of executing programs in a network", Johnson), as requested in a new parameter.
245	C	Moving a container causes searching the source system for the container. The container parameter can be well defined to narrow the search result, and may be wildcarded (pattern) for matching. In one embodiment, all occurrences found on the MS and their references/handles are provided with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In a preferred embodiment, the most recently accessed container meeting the search criteria is copied to the target systems. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In one embodiment, the move results in overwriting an existing container. In another embodiment, the move results in writing a newly altered reference/handle of the container when there is a conflict (e.g. existing entity with same name). An additional parameter may be specified for the target, otherwise the object is moved to an assumed location.

Fig. 71B-12

Operand ↓	PM	Preferred embodiment Move processing
247	C	Moving a program object causes first searching the source for the program object. In the preferred embodiment, a unique syntax is used for which type of program object is being sought (similar to above). There can be unique syntaxes for specifying which type of program object is being sought (e.g. "S:dataname"). In one embodiment, all occurrences found on the MS and information about the occurrence including its current value is presented to the user for reconciliation. In a preferred embodiment, one program object is to be found (e.g. >1 = ambiguous). In one embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Useful statistic(s) about the program object (perhaps which statistics specified with an additional parameter) are moved/copied to an appropriate destination of the target system(s) for informative purposes (e.g. a special log file). In another embodiment, the alerter process and/or indicator methodology can be used as the destination for the move for alerting a user at the target system. In another embodiment, there is new parameter for which end result the move will have (informative destination, handled like alert, handled like indicator). An alternate embodiment works like Operand 223 wherein moving is intended to keep program object(s) between systems in synch, albeit with a discard from the source system. Such embodiments require source and target system processing to have access to the object(s) (this may limit participating object(s)).
249	C	Moving a cursor causes searching the source system for the current cursor setting(s). In the preferred embodiment, provided in the search results is the current cursor information (e.g. image, animation , etc). The current cursor information of the source is then used to alter the cursor at the target system(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the move operation. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In some embodiments, the source cursor is reset to a different (e.g. initialized) setting as resulting from the move.

Fig. 71B-13

Operand ↓	PM	Preferred embodiment Move processing
251	C	Moving a calendar object causes searching the source calendar system with search criteria of the calendar object parameter string. The calendar object parameter string can specify searching any calendar entry fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria (similar to email above). Those skilled in the art recognize many useful syntaxes for searching any characteristics of calendar objects. In one embodiment, all occurrences found in history are presented to the user for reconciliation with at least their date/time stamps, attendees, and perhaps other information, of the calendar object and when it was scheduled. In a preferred embodiment, the most recent occurrence from the calendaring system is provided for the move. Wildcarding (pattern matching) is preferably inherent by searching for substrings. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The calendar entry is moved in its entirety to the target system calendaring system. In another embodiment, a new parameter is specified to move the calendar item(s) to a new schedule or time. A duplicate calendar entry may be created if one already exists.
253	C	Moving an address book (AB) object causes searching the source AB system with search criteria of the AB object parameter string. The AB object parameter string can specify searching any AB entry fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria (similar to email above). Those skilled in the art recognize many useful syntaxes for searching any characteristics of AB objects/entries. In one embodiment, all occurrences found are presented to the user for reconciliation with appropriate AB information. In a preferred embodiment, the most recent occurrence from the AB system is provided for the move. Wildcarding (pattern matching) is preferably inherent by searching for substrings. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The AB entry is moved in its entirety to the target system AB system. In another embodiment, a new parameter is specified to move the AB item(s) to a special destination. A duplicate AB entry may be created if one already exists.
. . .		

Fig. 71B-14

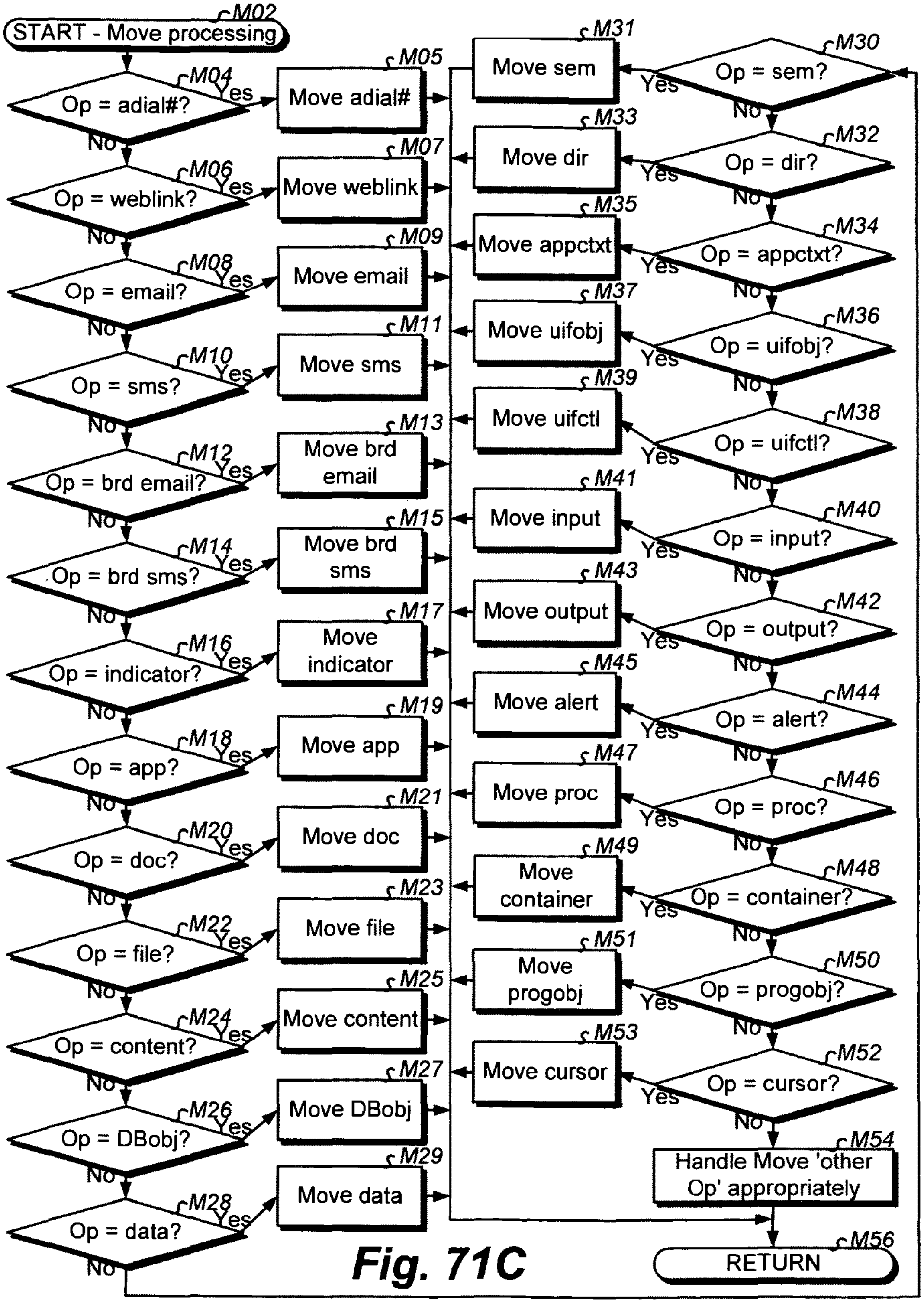

M02
START - Move processing
M04
Op = adial#?
Yes
No
M05
Move adial#
M06
Op = weblink?
M07
Move weblink
M08
Op = email?
M09
Move email
M10
Op = sms?
M11
Move sms
M12
Op = brd email?
M13
Move brd email
M14
Op = brd sms?
M15
Move brd sms
M16
Op = indicator?
M17
Move indicator
M18
Op = app?
M19
Move app
M20
Op = doc?
M21
Move doc
M22
Op = file?
M23
Move file
M24
Op = content?
M25
Move content
M26
Op = DBobj?
M27
Move DBobj
M28
Op = data?
M29
Move data
M30
Op = sem?
M31
Move sem
M32
Op = dir?
M33
Move dir
M34
Op = appctxt?
M35
Move appctxt
M36
Op = uifobj?
M37
Move uifobj
M38
Op = uifctl?
M39
Move uifctl
M40
Op = input?
M41
Move input
M42
Op = output?
M43
Move output
M44
Op = alert?
M45
Move alert
M46
Op = proc?
M47
Move proc
M48
Op = container?
M49
Move container
M50
Op = progobj?
M51
Move progobj
M52
Op = cursor?
M53
Move cursor
M54
Handle Move 'other Op' appropriately
M56
RETURN

Fig. 71C

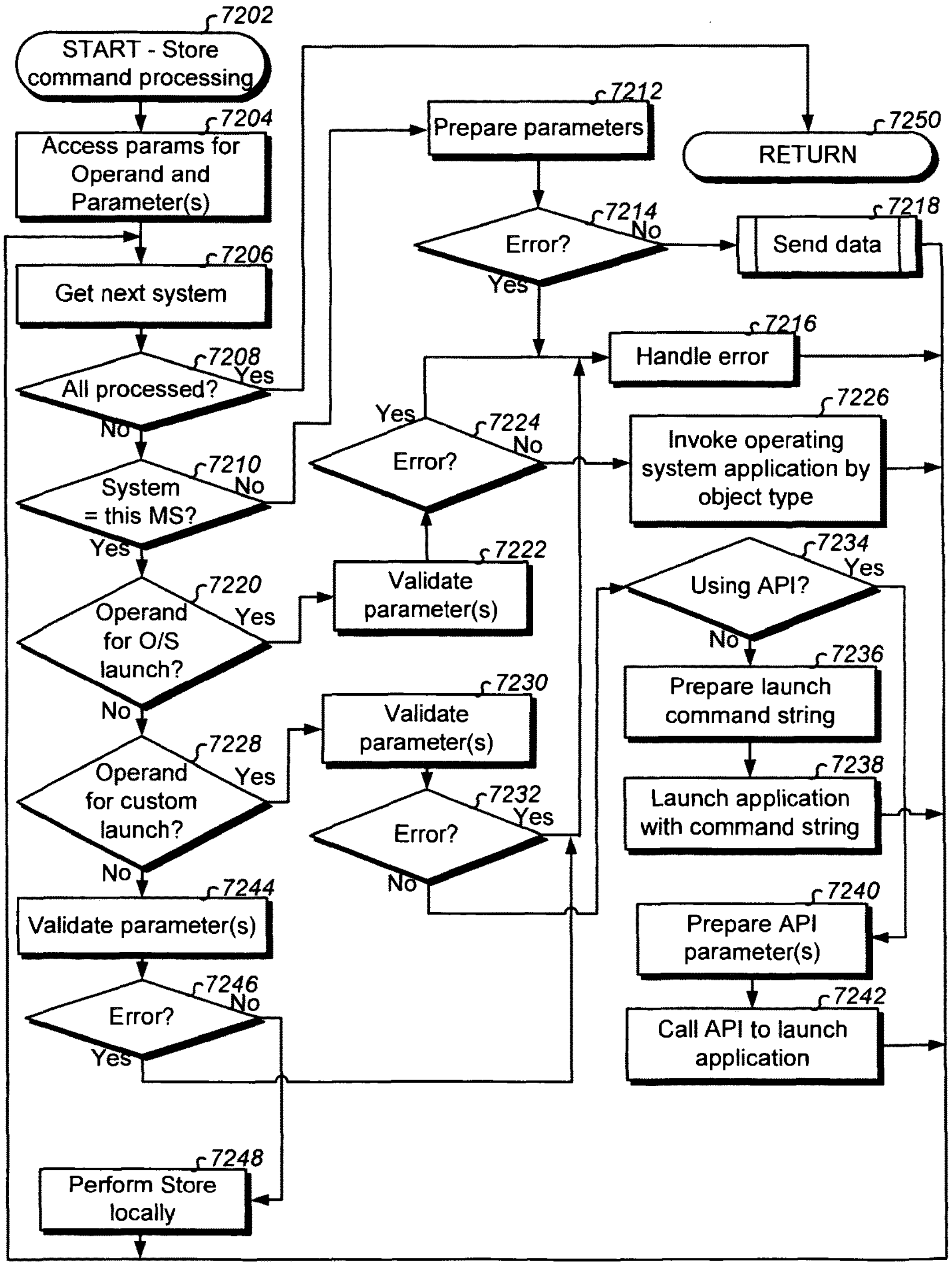

7202
START - Store command processing
7204
Access params for Operand and Parameter(s)
7206
Get next system
7208
All processed?
Yes
No
7210
System = this MS?
No
Yes
7220
Operand for O/S launch?
Yes
No
7228
Operand for custom launch?
Yes
No
7244
Validate parameter(s)
7246
Error?
No
Yes
7248
Perform Store locally
7212
Prepare parameters
7214
Error?
No
Yes
7218
Send data
7250
RETURN
7216
Handle error
7224
Error?
Yes
No
7222
Validate parameter(s)
7230
Validate parameter(s)
7232
Error?
Yes
No
7226
Invoke operating system application by object type
7234
Using API?
Yes
No
7236
Prepare launch command string
7238
Launch application with command string
7240
Prepare API parameter(s)
7242
Call API to launch application

Fig. 72A

Operand ↓	**PM**	**Preferred embodiment Store processing**
201	**C**	Storing an auto-dial # stores the auto-dial # parameter to the system(s). Preferably, a certain log is used to store the auto-dial #, or an additional parameter can be provided for where to store the auto-dial # to. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored auto-dial #. An additional parameter may be specified for how/where exactly to store it (e.g. which log).
203	**S**	Storing a weblink stores the weblink parameter to the system(s). Preferably, a certain link folder is used to store it, or an additional parameter can be provided for where to store it. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored weblink. An additional parameter may be specified for how/where exactly to store it (e.g. which folder).
205	**C**	Storing an email causes storing the email object to the system(s)' email system. In one embodiment, the email parameter string is a string containing a syntax for defining an email item and used to create the email to a certain folder, configured folder, or as specified with an additional parameter. Each email field can be defined with values, and the store command may result in defaulting fields which are not specified in the email parameter. In another embodiment, the email parameter points to a file containing a syntax for creating the email object. Those skilled in the art recognize many useful syntaxes for setting data in a new email object. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored email.

Fig. 72B-1

Operand ↓	PM	Preferred embodiment Store processing
207	C	Storing an sms message causes storing the sms message object to the system(s)' messaging system. In one embodiment, the sms message parameter string is a string containing a syntax for defining an sms message item and used to create the sms message to a certain folder, configured folder, or as specified with an additional parameter. Each sms message field can be defined with values, and the store command may result in defaulting fields which are not specified in the sms message parameter. In another embodiment, the sms message parameter points (e.g. path) to a file containing a syntax for creating the sms message object. Those skilled in the art recognize many useful syntaxes for setting data in a new sms message object. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored sms message.
209	C	Storing a broadcast email causes storing the email object to the system(s)' email system. In one embodiment, the email parameter string is a string containing a syntax for defining an email item and used to create the email to a certain folder, configured folder, or as specified with an additional parameter. Each email field can be defined with values, and the store command may result in defaulting fields which are not specified in the email parameter. In another embodiment, the email parameter points (e.g. path) to a file containing a syntax for creating the email object. Those skilled in the art recognize many useful syntaxes for setting data in a new email object. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored email.

Fig. 72B-2

Operand ↓	**PM**	**Preferred embodiment Store processing**
211	**C**	Storing a broadcast sms message causes storing the sms message object to the system(s)' messaging system. In one embodiment, the sms message parameter string is a string containing a syntax for defining an sms message item and used to create the sms message to a certain folder, configured folder, or as specified with an additional parameter. Each sms message field can be defined with values, and the store command may result in defaulting fields which are not specified in the sms message parameter. In another embodiment, the sms message parameter points to a file (e.g. path) containing a syntax for creating the sms message object. Those skilled in the art recognize many useful syntaxes for setting data in a new sms message object. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored sms message.
213	**O**	See Invoke Command for identical processing.
215	**C**	See Invoke Command for identical processing.
217	**S**	Storing a document stores the document parameter to the system(s). The document may be a self contained object parameter, or pointer (e.g. path) to a file defining the document. Preferably, a certain folder is used to store it, or an additional parameter can be provided for where to store it. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored document.
219	**S**	Storing a file stores the file from the path parameter to the system(s). Preferably, a certain folder is used to store it, or an additional parameter can be provided for where to store it. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored file.

Fig. 72B-3

Operand ↓	**PM**	**Preferred embodiment Store processing**
221	**O**	Storing content stores the content parameter to the system(s). The content may be a self contained object parameter, or pointer (e.g. path) to a file defining the content. Preferably, a certain destination is used to store it, or an additional parameter can be provided for where to store it. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored content.
223	**C**	See Invoke Command for identical processing.
225	**O**	See Invoke Command for identical processing.
227	**O**	See Invoke Command for identical processing.
229	**S**	Storing a directory stores the directory from the path parameter to the system(s). Preferably, a certain folder is used to store it, or an additional parameter can be provided for where to store it. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored directory.
231	**C**	See Invoke Command for identical processing.
233	**S**	Storing a focused user interface object causes a snapshot to be taken of the currently focused user interface object (to .jpg, .gif, alternate embodiments, etc) at the MS and then the snapshot file is stored as though the user manually captured the focused user interface object (e.g. Alt-Prtscrn) and saved it. Preferably, a certain folder is used to store it, or an additional parameter can be provided for where to store it. The first parameter command syntax can be defaulted or changed. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored snapshot. An additional parameter may be specified for the target format of the store whereby a conversion is caused (e.g. JPG to TIFF).
235	**O**	See Invoke Command for identical processing.
237	**O**	See Invoke Command for identical processing.
239	**O**	See Invoke Command for identical processing.

Fig. 72B-4

Operand ↓	PM	Preferred embodiment Store processing
241	S	Storing an alert causes storing the alert to the specified system(s). The alert parameter is the same parameter used to generate an alert (e.g. using another command). Storing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the store. Preferably, the store command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The store is performed so that the target system(s) are delivered the alert(s) like delivering a new alert to the systems.
243	O	Storing a process causes sending an operating system signal (see UNIX signaling) to the process name (after determining the Process ID (PID) of the prname parameter). A numeric value parameter (e.g. 0 or 1) may be communicated with the signal. An error is logged if the process is not found for signaling. Preferably, the store command data is maintained to LBX History, a log, or other useful storage for subsequent use.
245	S	Storing a container stores the container parameter to the system(s). The container may be a self contained object parameter, or pointer (e.g. path) to a file defining the container. Preferably, a certain destination is used to store it, or an additional parameter can be provided for where to store it. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored container.
247	O	See Invoke Command for identical processing.
249	O	See Invoke Command for identical processing.
251	C	See Invoke Command for identical processing.
253	C	See Invoke Command for identical processing.
. . .		

Fig. 72B-5

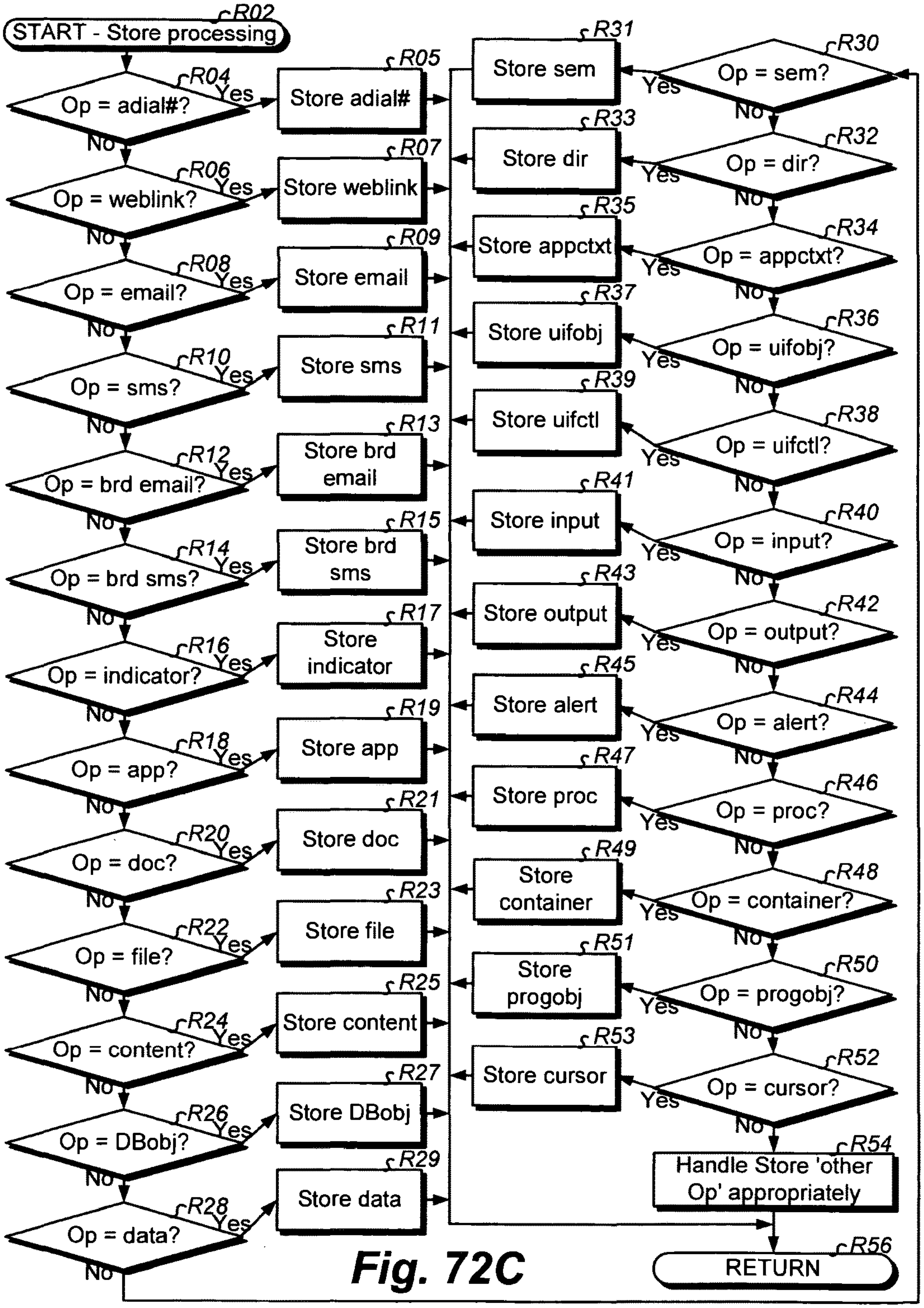

START - Store processing
R02
Op = adial#?
R04
Yes
No
Store adial#
R05
Op = weblink?
R06
Store weblink
R07
Op = email?
R08
Store email
R09
Op = sms?
R10
Store sms
R11
Op = brd email?
R12
Store brd email
R13
Op = brd sms?
R14
Store brd sms
R15
Op = indicator?
R16
Store indicator
R17
Op = app?
R18
Store app
R19
Op = doc?
R20
Store doc
R21
Op = file?
R22
Store file
R23
Op = content?
R24
Store content
R25
Op = DBobj?
R26
Store DBobj
R27
Op = data?
R28
Store data
R29
Op = sem?
R30
Store sem
R31
Op = dir?
R32
Store dir
R33
Op = appctxt?
R34
Store appctxt
R35
Op = uifobj?
R36
Store uifobj
R37
Op = uifctl?
R38
Store uifctl
R39
Op = input?
R40
Store input
R41
Op = output?
R42
Store output
R43
Op = alert?
R44
Store alert
R45
Op = proc?
R46
Store proc
R47
Op = container?
R48
Store container
R49
Op = progobj?
R50
Store progobj
R51
Op = cursor?
R52
Store cursor
R53
Handle Store 'other Op' appropriately
R54
RETURN
R56

Fig. 72C

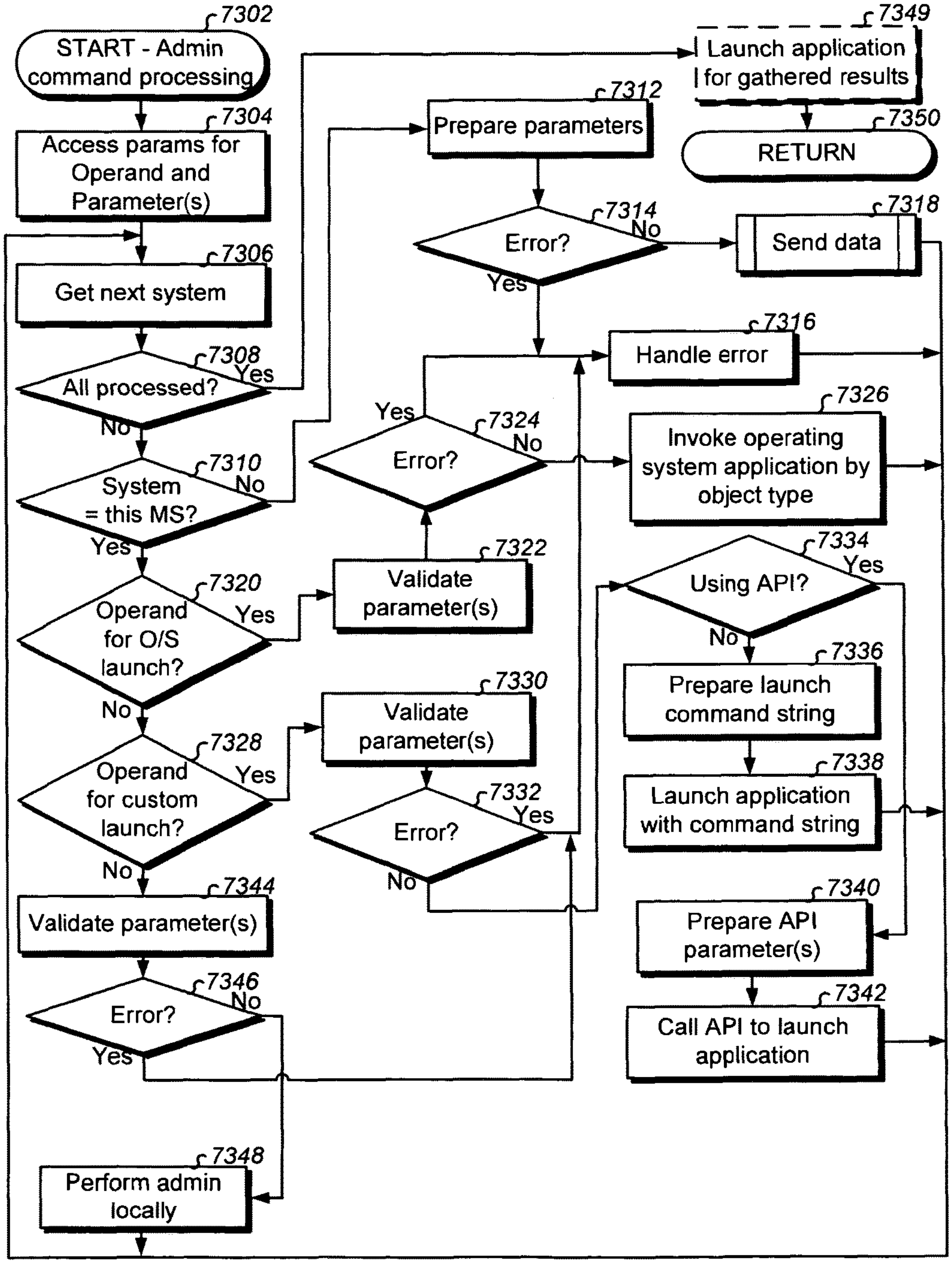

7302
START - Admin command processing
7304
Access params for Operand and Parameter(s)
7306
Get next system
7308
All processed?
Yes
No
7310
System = this MS?
No
Yes
7320
Operand for O/S launch?
Yes
No
7328
Operand for custom launch?
Yes
No
7344
Validate parameter(s)
7346
Error?
No
Yes
7348
Perform admin locally
7312
Prepare parameters
7314
Error?
No
Yes
7318
Send data
7316
Handle error
7324
Error?
Yes
No
7322
Validate parameter(s)
7330
Validate parameter(s)
7332
Error?
Yes
No
7349
Launch application for gathered results
7350
RETURN
7326
Invoke operating system application by object type
7334
Using API?
Yes
No
7336
Prepare launch command string
7338
Launch application with command string
7340
Prepare API parameter(s)
7342
Call API to launch application

Fig. 73A

Operand ↓	PM	Preferred embodiment Administrate processing
201	**C**	Administrating an auto-dial # launches a MS phone number log interface with the auto-dial # parameter for searching. Preferably, both the outgoing and incoming logs are searched. The auto-dial # parameter can be a wildcard (pattern) for matching. In one embodiment, all occurrences found in history are presented with their date/time stamps, log found in, and perhaps other information, of the call and when it took place. In another embodiment, the most recent occurrence from a particular log is presented, and perhaps in an interface which enables calling the # with a minimal user action. The search takes place as though the user manually launched the search, entered the auto-dial # for the search, and then was presented with the result(s) in an appropriate administration interface. Appropriate MS storage is updated and subsequently processed as though the user manually performed the search. Preferably, the administration cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Subsequent administration includes modifying the log entry(s) as desired (e.g. add a 1 prefix since caller id may not have maintained one when it is needed for auto-dial). A new parameter can be specified for which log(s) to search.
203	**S**	Administrating a weblink launches a search to MS browser history with the weblink parameter (and with the params parameter if specified) for searching. The weblink parameter can be a wildcard (pattern) for matching. In one embodiment, all occurrences found in history are presented with their date/time stamps, folder found in, and perhaps other information, of the link and when it was invoked. In another embodiment, the most recent occurrence from a particular invocation is presented, and perhaps in an interface which enables invoking (transposing to) the weblink with a minimal user action. The search takes place as though the user manually launched the search, entered the weblink for the search, and then was presented with the result(s) in an appropriate administration interface. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the administration command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Subsequent administration includes modifying the weblink(s) as desired (e.g. change description).

Fig. 73B-1

Operand ↓	**PM**	**Preferred embodiment Administrate processing**
205	**C**	Administrating an email causes searching a MS email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding (patterns for matching). Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel';recip:'george@alltell.com';body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. All occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS email system processing from that point forward (e.g. when processed at local MS). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s) in an appropriate administration interface. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the administration command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Subsequent administration includes modifying the email(s) as desired, and perhaps having an option to send/resend.

Fig. 73B-2

Operand ↓	PM	Preferred embodiment Administrate processing
207	C	Administrating an sms message causes searching a MS sms messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In the preferred embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS messaging system processing from that point forward (e.g. when processed at local MS). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s) in an appropriate administration interface. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the administration command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Subsequent administration includes modifying the sms message(s) as desired, and perhaps having an option to send/resend.

Fig. 73B-3

Operand ↓	**PM**	**Preferred embodiment Administrate processing**
209	**C**	Administrating a broadcast email causes searching a MS email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel';recip:'george@alltell.com';body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In the preferred embodiment, all occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS email system processing from that point forward (e.g. when processed at local MS). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s) in an appropriate administration interface. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the administration command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Subsequent administration includes modifying the email(s) as desired, and perhaps having an option to send/resend.

Fig. 73B-4

Operand ↓	PM	Preferred embodiment Administrate processing
211	C	Administrating a broadcast sms message causes searching a MS sms messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In the preferred embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS messaging system processing from that point forward (e.g. when processed at local MS). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s) in an appropriate administration interface. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the administration command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Subsequent administration includes modifying the message(s) as desired, and perhaps having an option to send/resend.
213	O	See Find Command for identical processing this LBX release.

Fig. 73B-5

Operand ↓	PM	Preferred embodiment Administrate processing
215	C	Administrating an application causes searching the MS for application (and with the params parameter if specified). The app parameter is preferably an executable name. Providing a more defined partial or full path to the application parameter will validate that it is found there. The app parameter string preferably supports wildcarding. In the preferred embodiment, all occurrences found on the MS and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information. In another embodiment, all parts which are linked to the executable are identified with their paths, date/time stamps, size, and perhaps attributes when a symbol file is specified with a new parameter. The symbol file is output from a link process and can be used to identify all executable parts such as dynamic link libraries, linked binaries, and any other executable binary file involved with the application. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s) in an appropriate administration interface (e.g. a properties edit user interface). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the administration command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Subsequent administration includes modifying the configuration, startup parameters, or any other environmental variables of the application.
217	S	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
219	S	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
221	O	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
223	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
225	O	See Invoke Command for identical processing this LBX release.
227	O	See Invoke Command for identical processing this LBX release.
229	S	See Invoke Command for identical processing this LBX release.
231	C	See Invoke Command for identical processing this LBX release.
233	S	See Invoke Command for identical processing this LBX release.
235	O	See Invoke Command for identical processing this LBX release.
237	O	See Invoke Command for identical processing this LBX release.
239	O	See Invoke Command for identical processing this LBX release.

Fig. 73B-6

Operand ↓	PM	Preferred embodiment Administrate processing
241	S	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
243	O	See Invoke Command for identical processing this LBX release.
245	S	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
247	O	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
249	O	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
251	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
253	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
. . .		

Fig. 73B-7

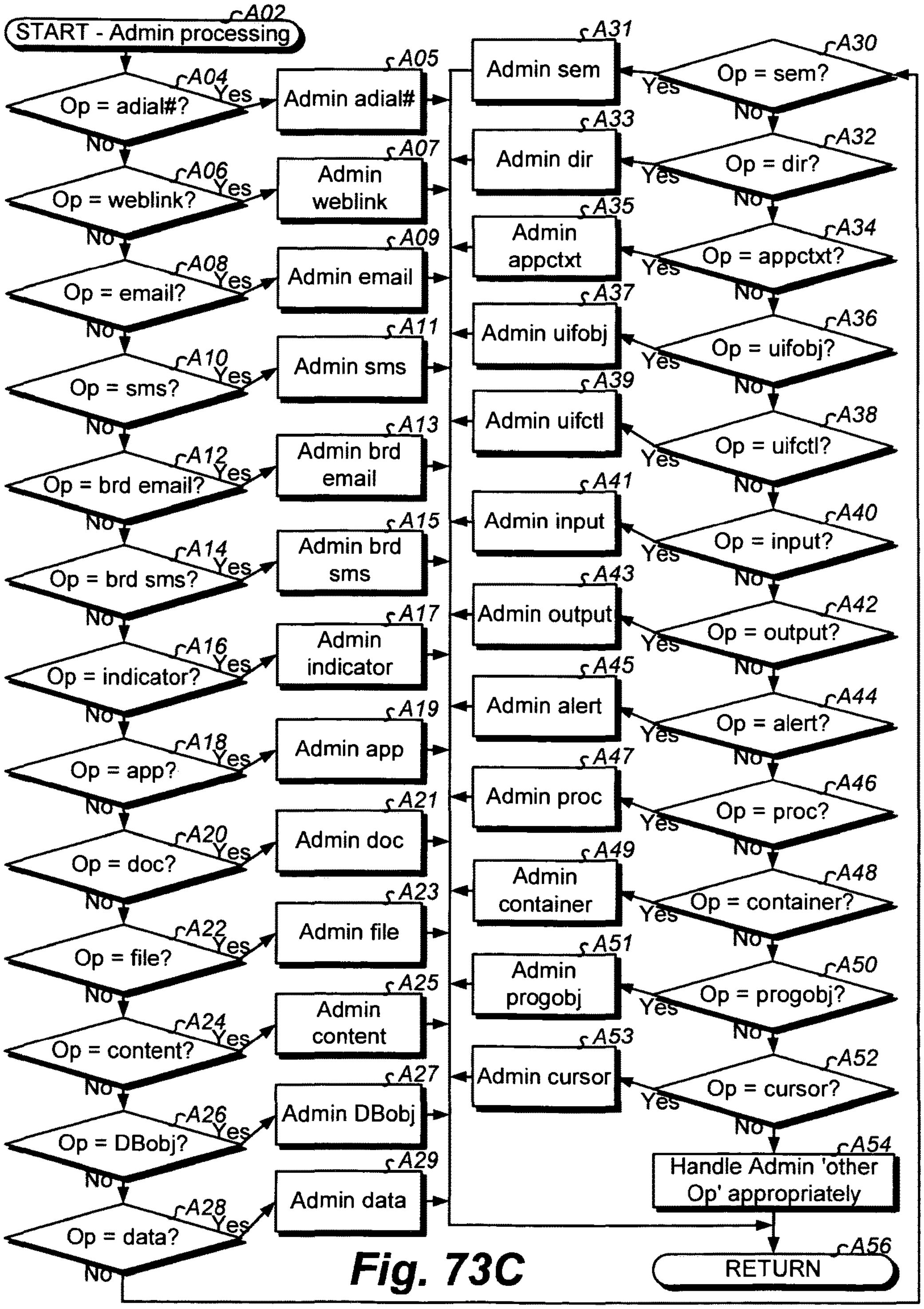

START - Admin processing
A02
Op = adial#?
A04
Admin adial#
A05
Op = weblink?
A06
Admin weblink
A07
Op = email?
A08
Admin email
A09
Op = sms?
A10
Admin sms
A11
Op = brd email?
A12
Admin brd email
A13
Op = brd sms?
A14
Admin brd sms
A15
Op = indicator?
A16
Admin indicator
A17
Op = app?
A18
Admin app
A19
Op = doc?
A20
Admin doc
A21
Op = file?
A22
Admin file
A23
Op = content?
A24
Admin content
A25
Op = DBobj?
A26
Admin DBobj
A27
Op = data?
A28
Admin data
A29
Op = sem?
A30
Admin sem
A31
Op = dir?
A32
Admin dir
A33
Op = appctxt?
A34
Admin appctxt
A35
Op = uifobj?
A36
Admin uifobj
A37
Op = uifctl?
A38
Admin uifctl
A39
Op = input?
A40
Admin input
A41
Op = output?
A42
Admin output
A43
Op = alert?
A44
Admin alert
A45
Op = proc?
A46
Admin proc
A47
Op = container?
A48
Admin container
A49
Op = progobj?
A50
Admin progobj
A51
Op = cursor?
A52
Admin cursor
A53
Handle Admin 'other Op' appropriately
A54
RETURN
A56
Yes
No

Fig. 73C

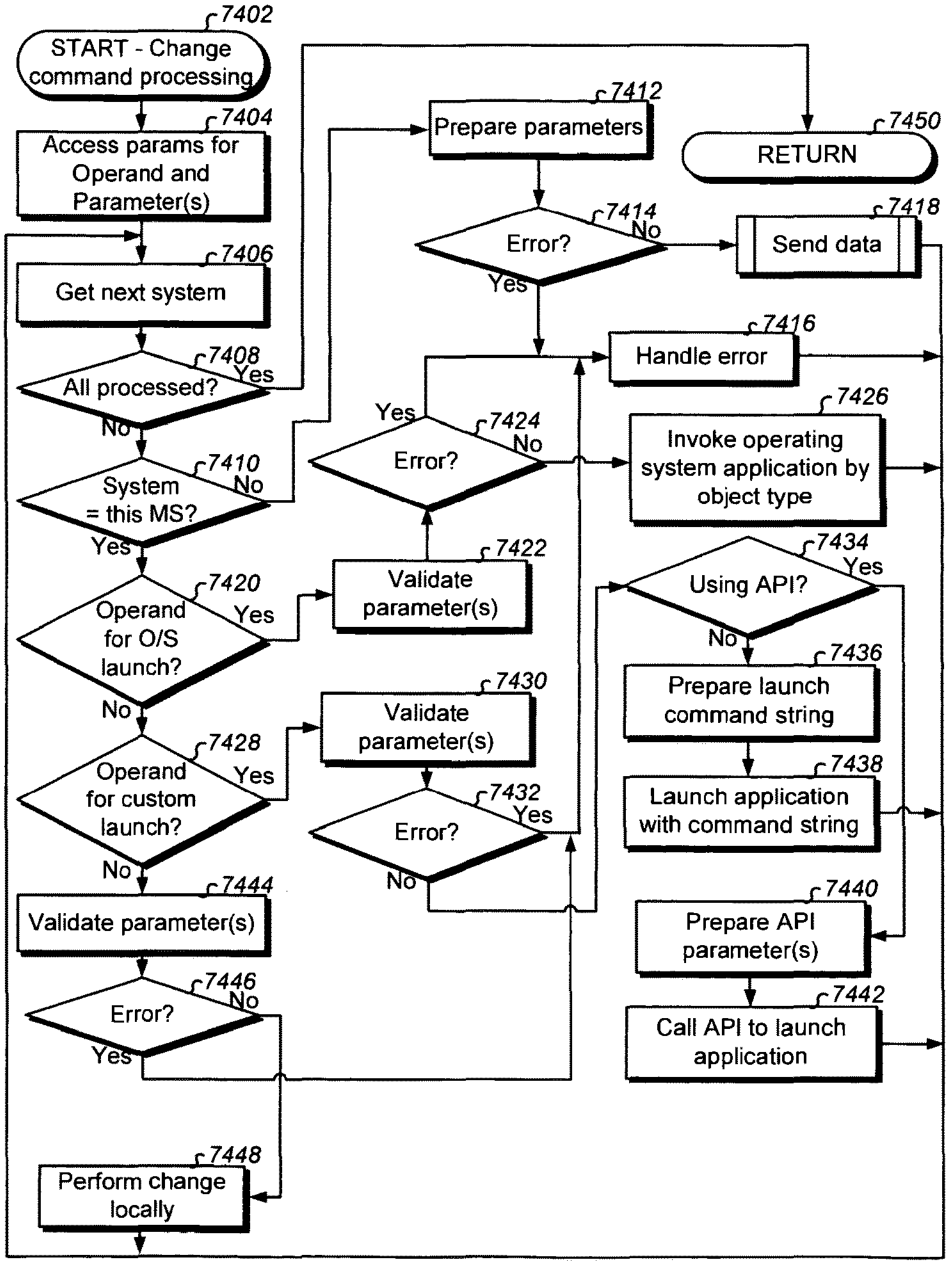

7402
START - Change command processing
7404
Access params for Operand and Parameter(s)
7406
Get next system
7408
All processed?
Yes
No
7410
System = this MS?
No
Yes
7420
Operand for O/S launch?
Yes
No
7428
Operand for custom launch?
Yes
No
7444
Validate parameter(s)
7446
Error?
No
Yes
7448
Perform change locally
7412
Prepare parameters
7414
Error?
No
Yes
7418
Send data
7450
RETURN
7416
Handle error
7424
Error?
Yes
No
7422
Validate parameter(s)
7430
Validate parameter(s)
7432
Error?
Yes
No
7426
Invoke operating system application by object type
7434
Using API?
Yes
No
7436
Prepare launch command string
7438
Launch application with command string
7440
Prepare API parameter(s)
7442
Call API to launch application

Fig. 74A

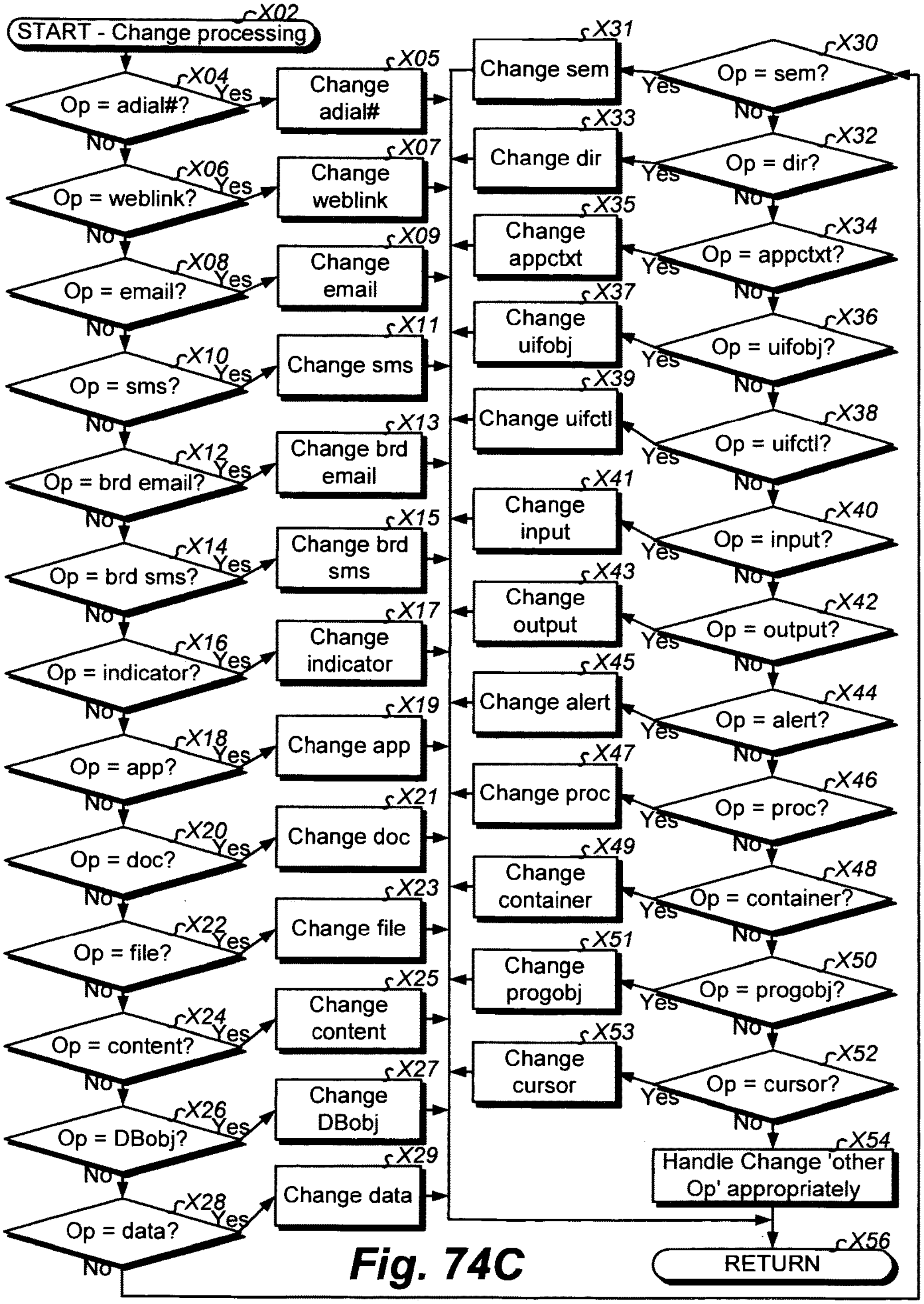

X02
START - Change processing
X04
Op = adial#?
Yes
No
X05
Change adial#
X06
Op = weblink?
X07
Change weblink
X08
Op = email?
X09
Change email
X10
Op = sms?
X11
Change sms
X12
Op = brd email?
X13
Change brd email
X14
Op = brd sms?
X15
Change brd sms
X16
Op = indicator?
X17
Change indicator
X18
Op = app?
X19
Change app
X20
Op = doc?
X21
Change doc
X22
Op = file?
X23
Change file
X24
Op = content?
X25
Change content
X26
Op = DBobj?
X27
Change DBobj
X28
Op = data?
X29
Change data
X30
Op = sem?
X31
Change sem
X32
Op = dir?
X33
Change dir
X34
Op = appctxt?
X35
Change appctxt
X36
Op = uifobj?
X37
Change uifobj
X38
Op = uifctl?
X39
Change uifctl
X40
Op = input?
X41
Change input
X42
Op = output?
X43
Change output
X44
Op = alert?
X45
Change alert
X46
Op = proc?
X47
Change proc
X48
Op = container?
X49
Change container
X50
Op = progobj?
X51
Change progobj
X52
Op = cursor?
X53
Change cursor
X54
Handle Change 'other Op' appropriately
X56
RETURN

Fig. 74C

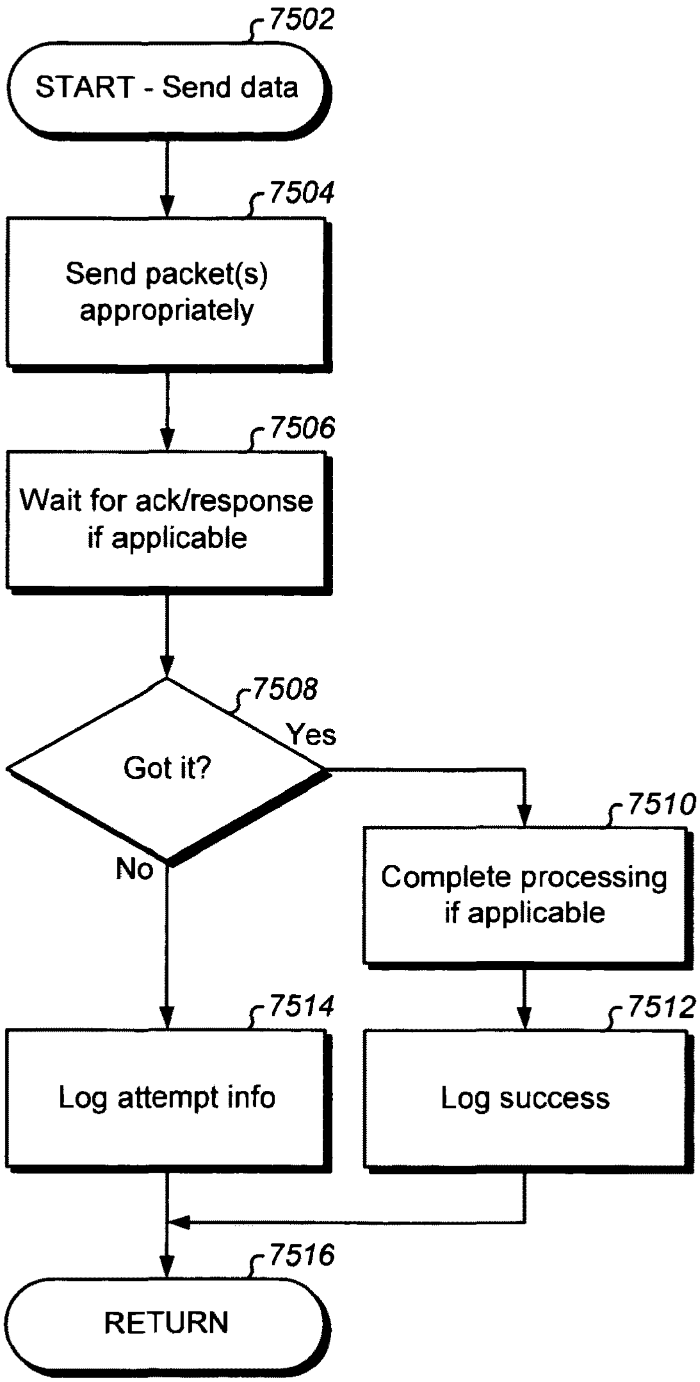

7502
START - Send data
7504
Send packet(s) appropriately
7506
Wait for ack/response if applicable
7508
Got it?
Yes
No
7510
Complete processing if applicable
7512
Log success
7514
Log attempt info
7516
RETURN

Fig. 75A

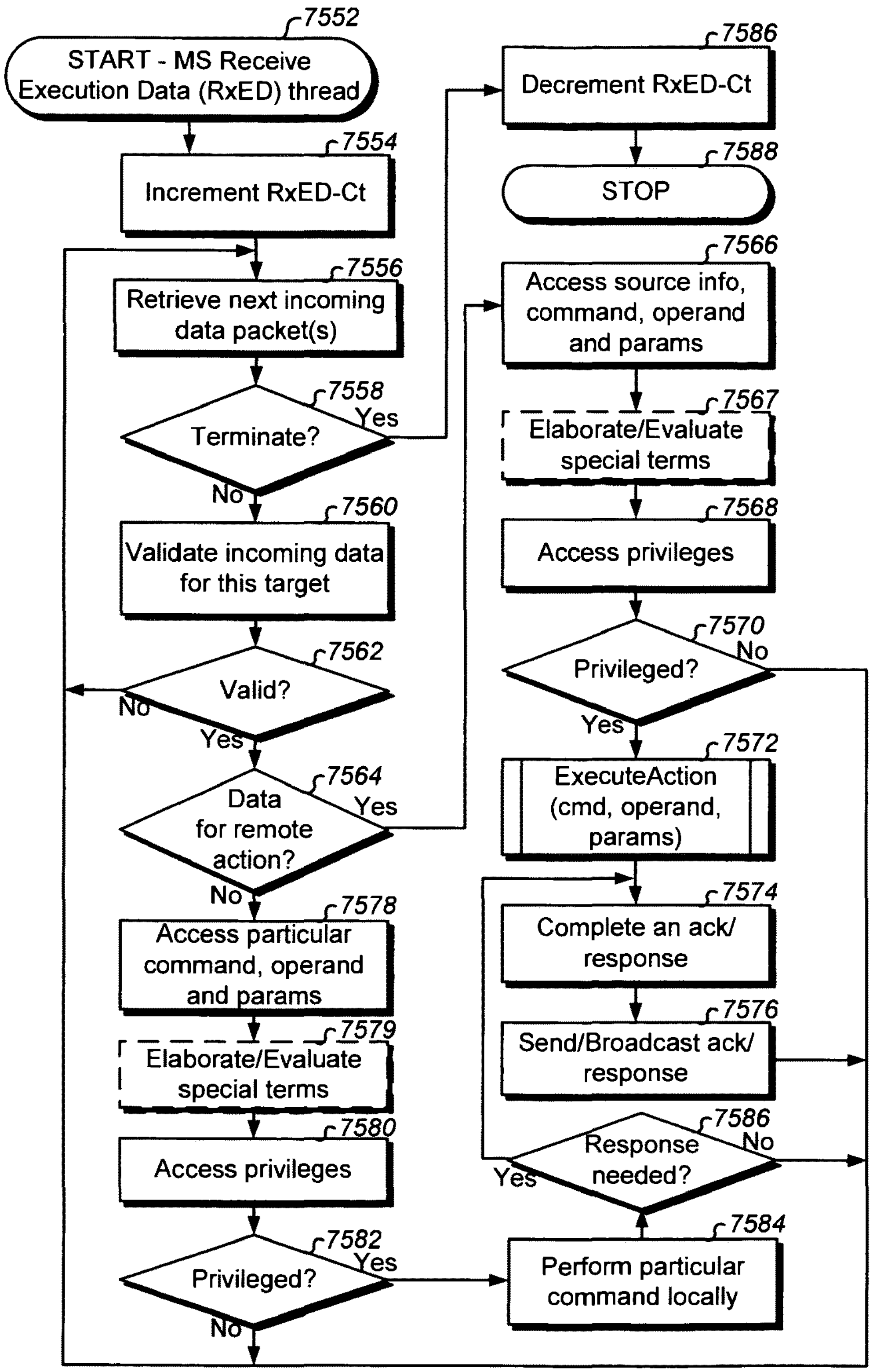

7552
START - MS Receive Execution Data (RxED) thread
7554
Increment RxED-Ct
7556
Retrieve next incoming data packet(s)
7558
Terminate?
Yes
No
7560
Validate incoming data for this target
7562
Valid?
No
Yes
7564
Data for remote action?
Yes
No
7578
Access particular command, operand and params
7579
Elaborate/Evaluate special terms
7580
Access privileges
7582
Privileged?
Yes
No
7586
Decrement RxED-Ct
7588
STOP
7566
Access source info, command, operand and params
7567
Elaborate/Evaluate special terms
7568
Access privileges
7570
Privileged?
No
Yes
7572
ExecuteAction (cmd, operand, params)
7574
Complete an ack/ response
7576
Send/Broadcast ack/ response
7586
Response needed?
No
Yes
7584
Perform particular command locally

Fig. 75B

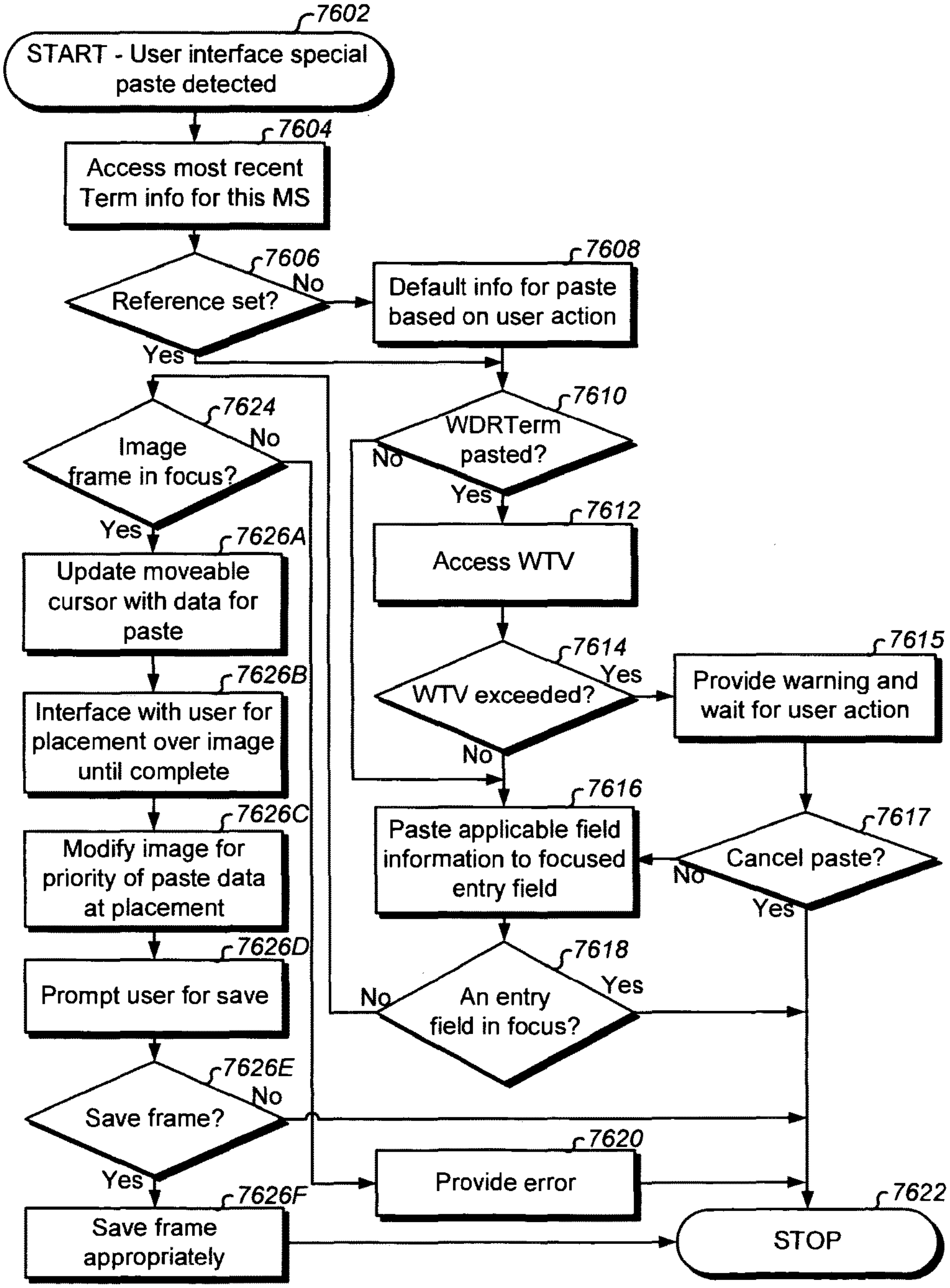

7602
START - User interface special paste detected
7604
Access most recent Term info for this MS
7606
Reference set?
No
Yes
7608
Default info for paste based on user action
7610
WDRTerm pasted?
No
Yes
7612
Access WTV
7614
WTV exceeded?
Yes
No
7615
Provide warning and wait for user action
7617
Cancel paste?
No
Yes
7616
Paste applicable field information to focused entry field
7618
An entry field in focus?
No
Yes
7624
Image frame in focus?
No
Yes
7626A
Update moveable cursor with data for paste
7626B
Interface with user for placement over image until complete
7626C
Modify image for priority of paste data at placement
7626D
Prompt user for save
7626E
Save frame?
No
Yes
7626F
Save frame appropriately
7620
Provide error
7622
STOP

Fig. 76A

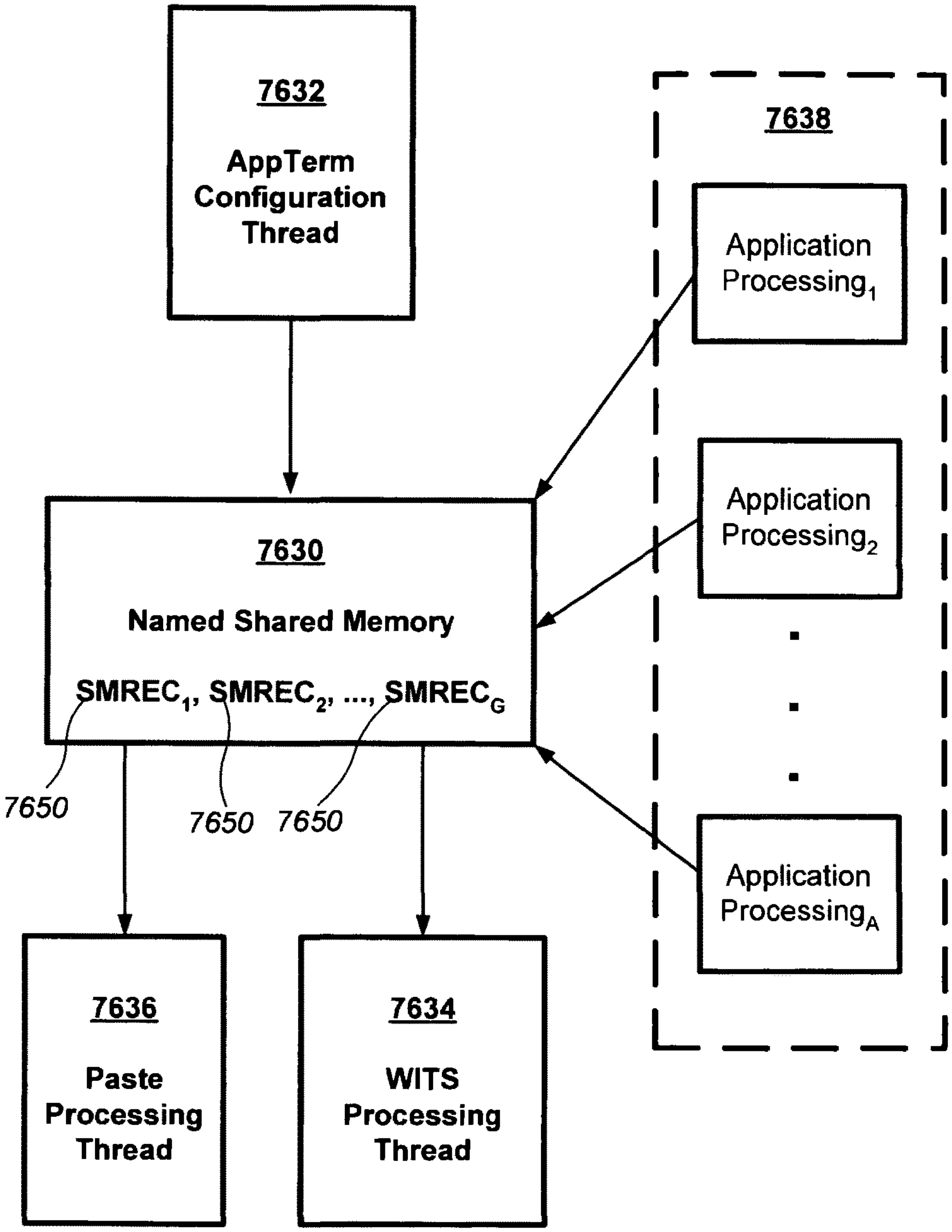

7632
AppTerm Configuration Thread
7638
Application Processing$_1$
Application Processing$_2$
Application Processing$_A$
7630
Named Shared Memory
SMREC$_1$, SMREC$_2$, ..., SMREC$_G$
7650
7650
7650
7636
Paste Processing Thread
7634
WITS Processing Thread

Fig. 76B-1

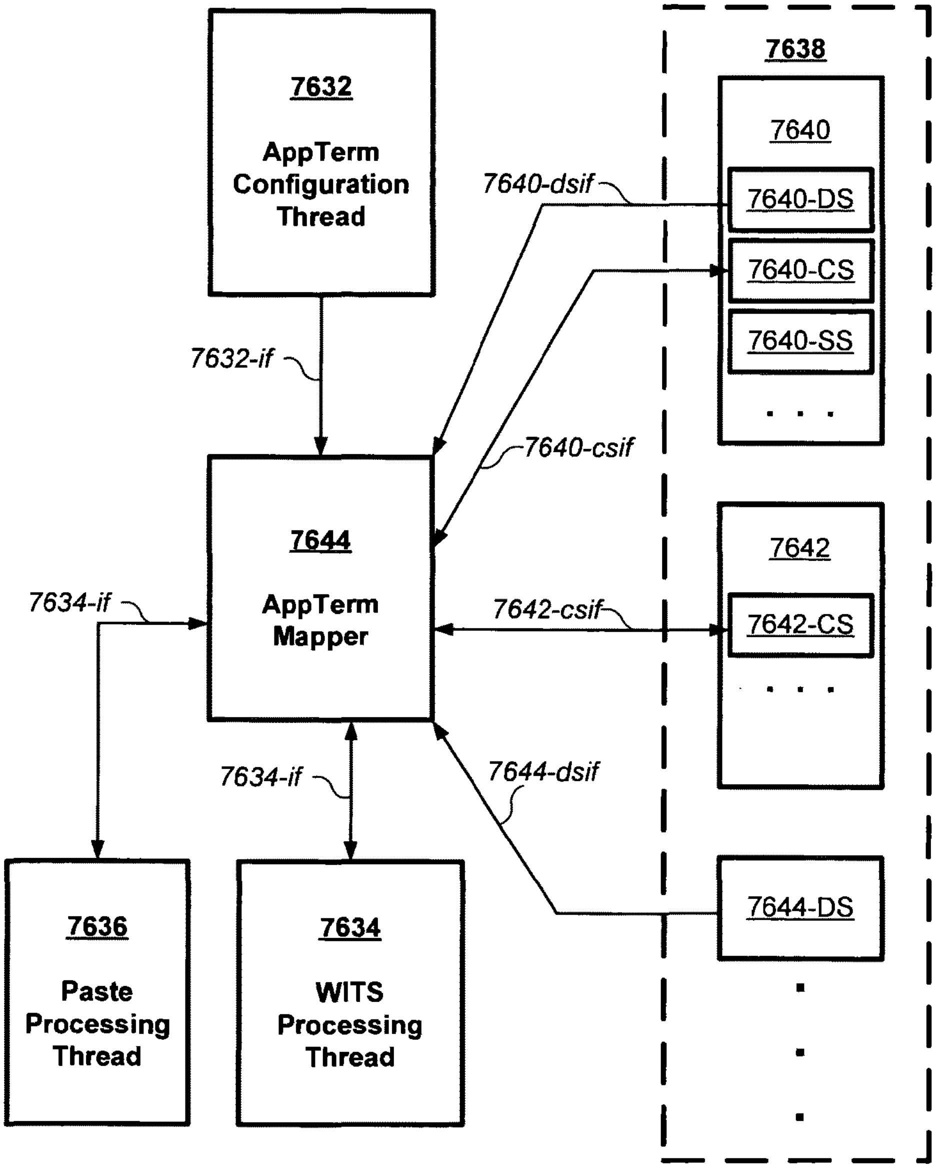

7632
AppTerm Configuration Thread
7632-if
7644
AppTerm Mapper
7634-if
7634-if
7636
Paste Processing Thread
7634
WITS Processing Thread
7638
7640
7640-DS
7640-CS
7640-SS
7640-dsif
7640-csif
7642
7642-CS
7642-csif
7644-dsif
7644-DS

Fig. 76B-2

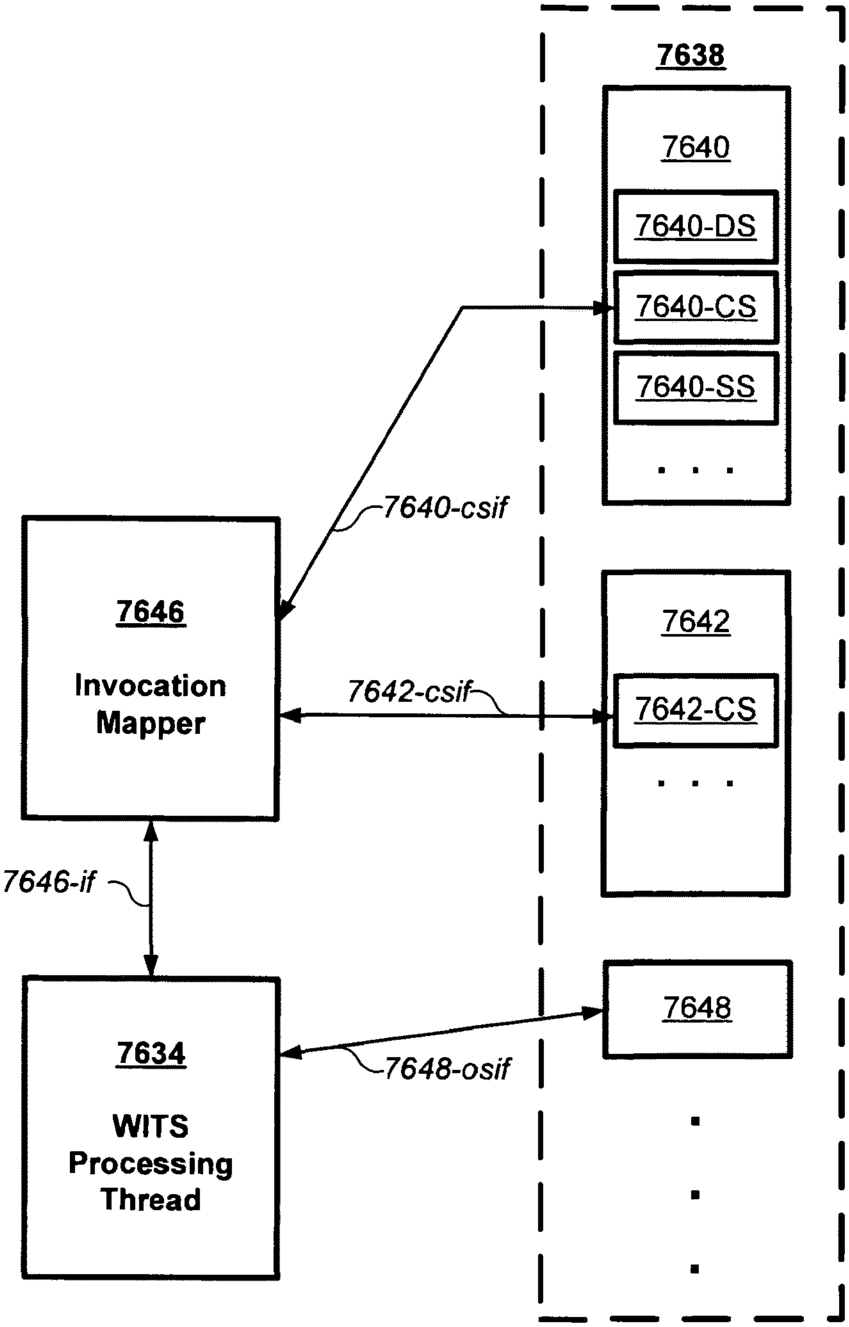

7638
7640
7640-DS
7640-CS
7640-SS
. . .
7640-csif
7646
Invocation Mapper
7642
7642-csif
7642-CS
. . .
7646-if
7634
WITS Processing Thread
7648
7648-osif
.
.
.

Fig. 76B-3

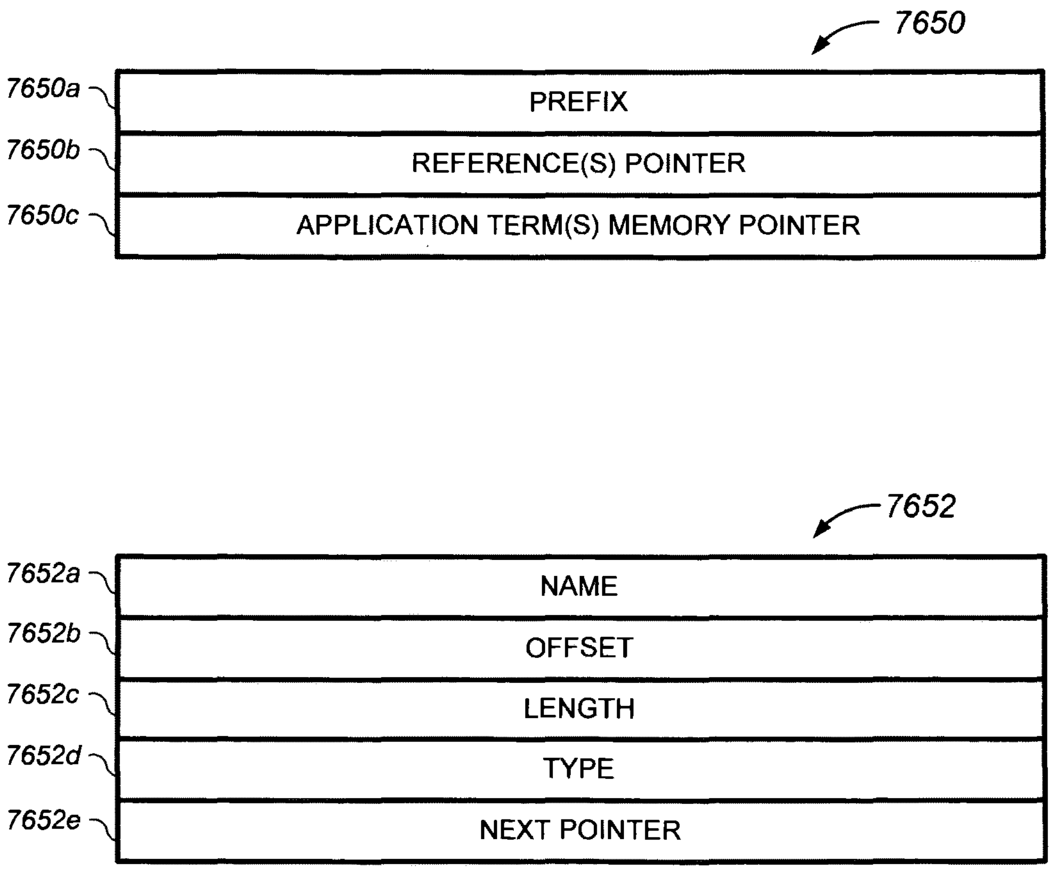

7650
7650a
PREFIX
7650b
REFERENCE(S) POINTER
7650c
APPLICATION TERM(S) MEMORY POINTER
7652
7652a
NAME
7652b
OFFSET
7652c
LENGTH
7652d
TYPE
7652e
NEXT POINTER

Fig. 76C

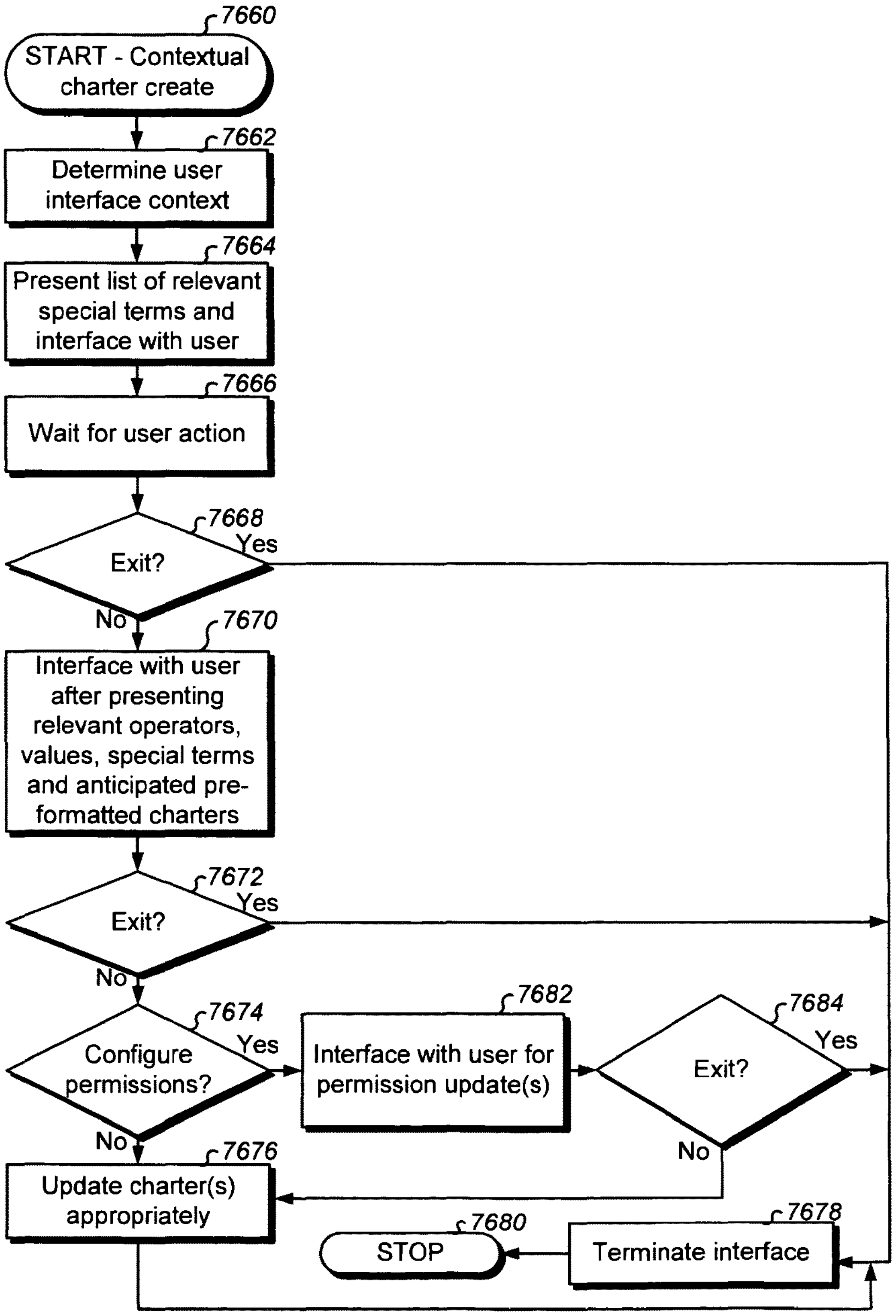

7660
START - Contextual charter create
7662
Determine user interface context
7664
Present list of relevant special terms and interface with user
7666
Wait for user action
7668
Exit?
Yes
No
7670
Interface with user after presenting relevant operators, values, special terms and anticipated pre-formatted charters
7672
Exit?
Yes
No
7674
Configure permissions?
Yes
No
7682
Interface with user for permission update(s)
7684
Exit?
Yes
No
7676
Update charter(s) appropriately
7680
STOP
7678
Terminate interface

Fig. 76D

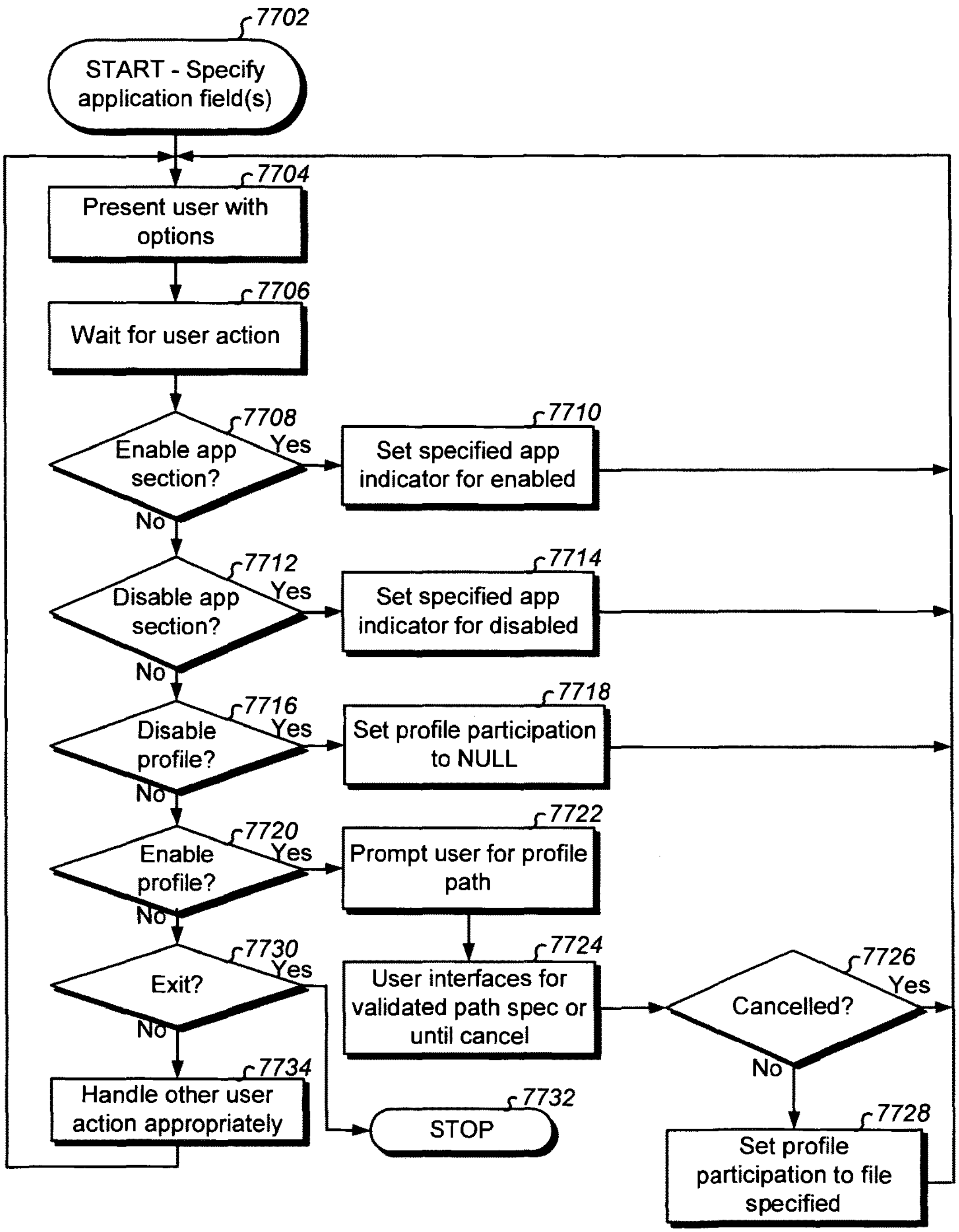
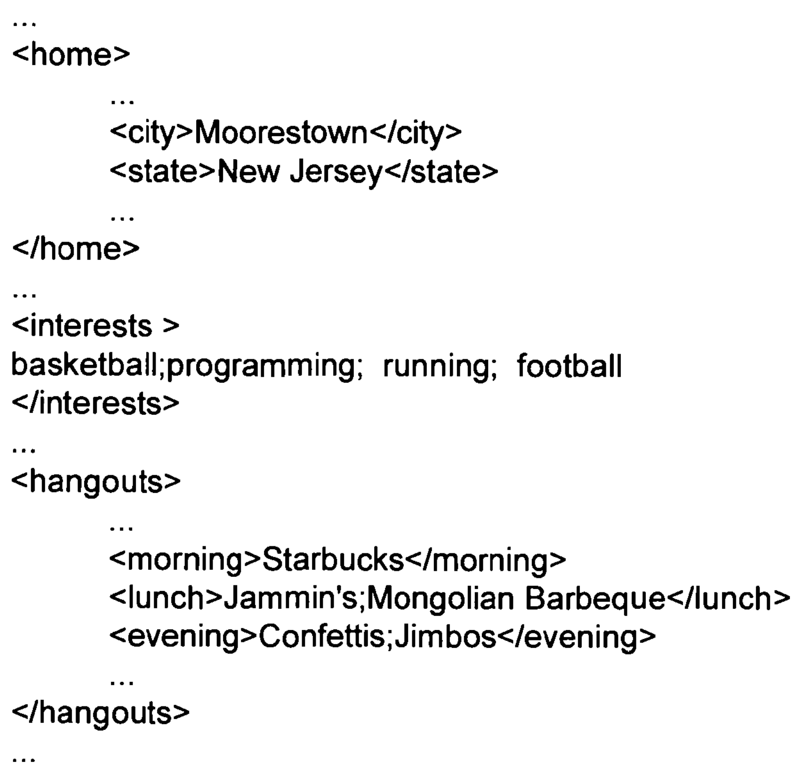

7702
START - Specify application field(s)
7704
Present user with options
7706
Wait for user action
7708
Enable app section?
Yes
No
7710
Set specified app indicator for enabled
7712
Disable app section?
Yes
No
7714
Set specified app indicator for disabled
7716
Disable profile?
Yes
No
7718
Set profile participation to NULL
7720
Enable profile?
Yes
No
7722
Prompt user for profile path
7730
Exit?
Yes
No
7724
User interfaces for validated path spec or until cancel
7726
Cancelled?
Yes
No
7734
Handle other user action appropriately
7732
STOP
7728
Set profile participation to file specified

Fig. 77



```
...
<home>
        ...
        <city>Moorestown</city>
        <state>New Jersey</state>
        ...
</home>
...
<interests >
basketball;programming;  running;  football
</interests>
...
<hangouts>
        ...
        <morning>Starbucks</morning>
        <lunch>Jammin's;Mongolian Barbeque</lunch>
        <evening>Confettis;Jimbos</evening>
        ...
</hangouts>
...
```

Fig. 78

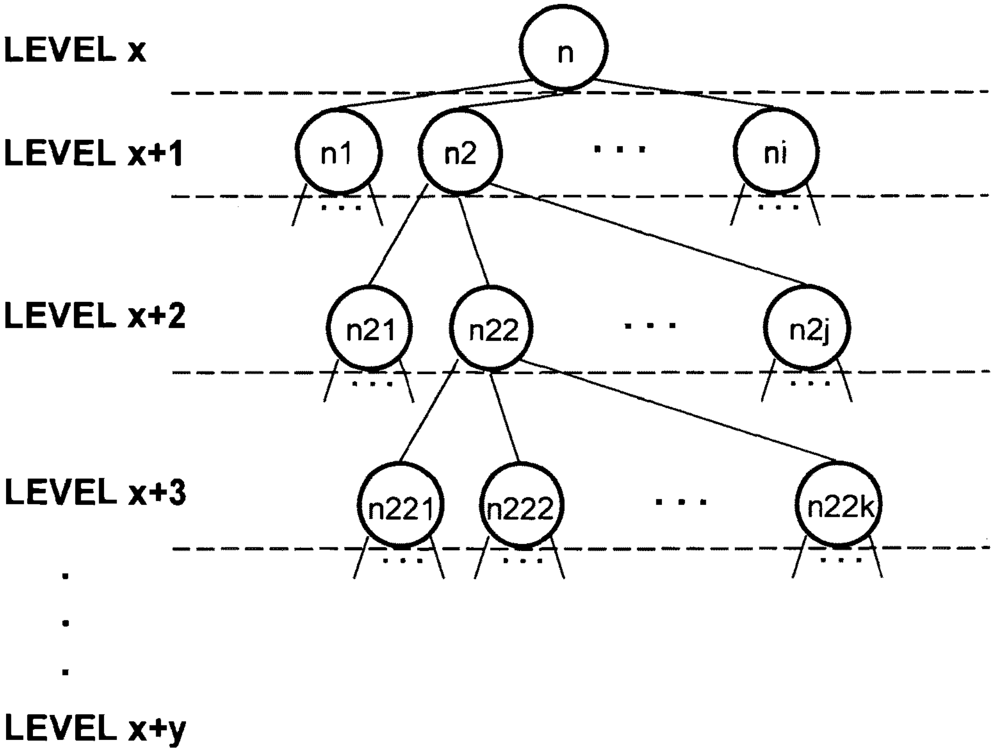

LEVEL x
LEVEL x+1
LEVEL x+2
LEVEL x+3
.
.
.
LEVEL x+y
n
n1
n2
. . .
ni
n21
n22
. . .
n2j
n221
n222
. . .
n22k

Fig. 79A

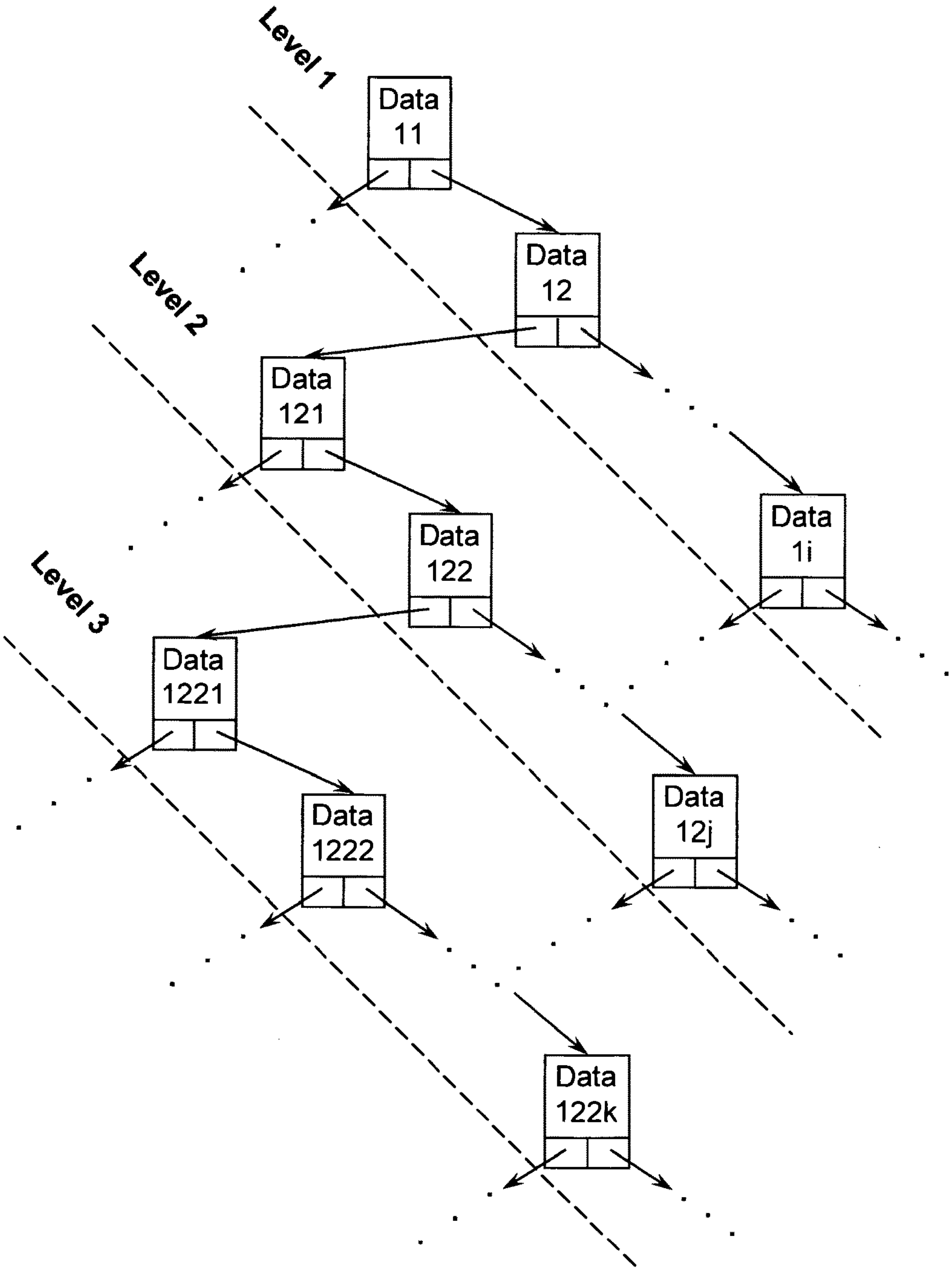

Level 1
Level 2
Level 3
Data 11
Data 12
Data 121
Data 122
Data 1i
Data 1221
Data 1222
Data 12j
Data 122k

Fig. 79B

```
.
.
.
typedef struct xml_node {
                struct xml_node    *descend1st; // Points to first descendant tag in XML doc
                struct xml_node    *peer_down;  // Points to same level tag down XML doc
                struct xml_node    *ascendant;  // Points to ascending tag in XML doc
                struct xml_node    *peer_up;    // Points to same level tag up XML doc
                char                data_type;  // Type of data @ data pointer
                unsigned char      *data;       // Typecast-able pointer to data
                } XML_NODE;
.
.
.
XML_NODE  *Lprofile;     // Root node to XML doc tag tree for Local profile
.
.
.
XML_NODE  *Rprofile;     // Root node to XML doc tag tree for Remote/Received profile
.
.
.
```

Fig. 79C

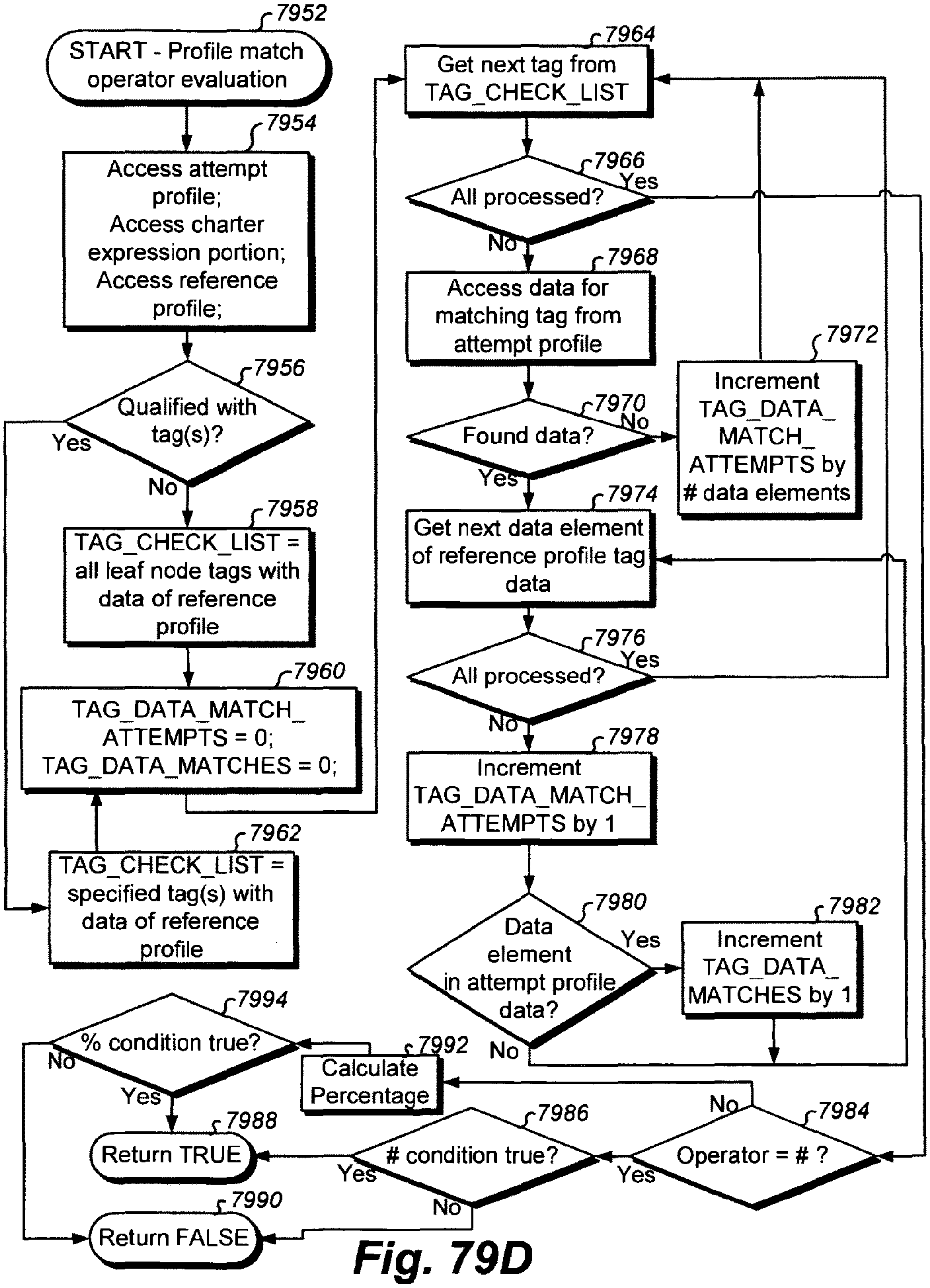
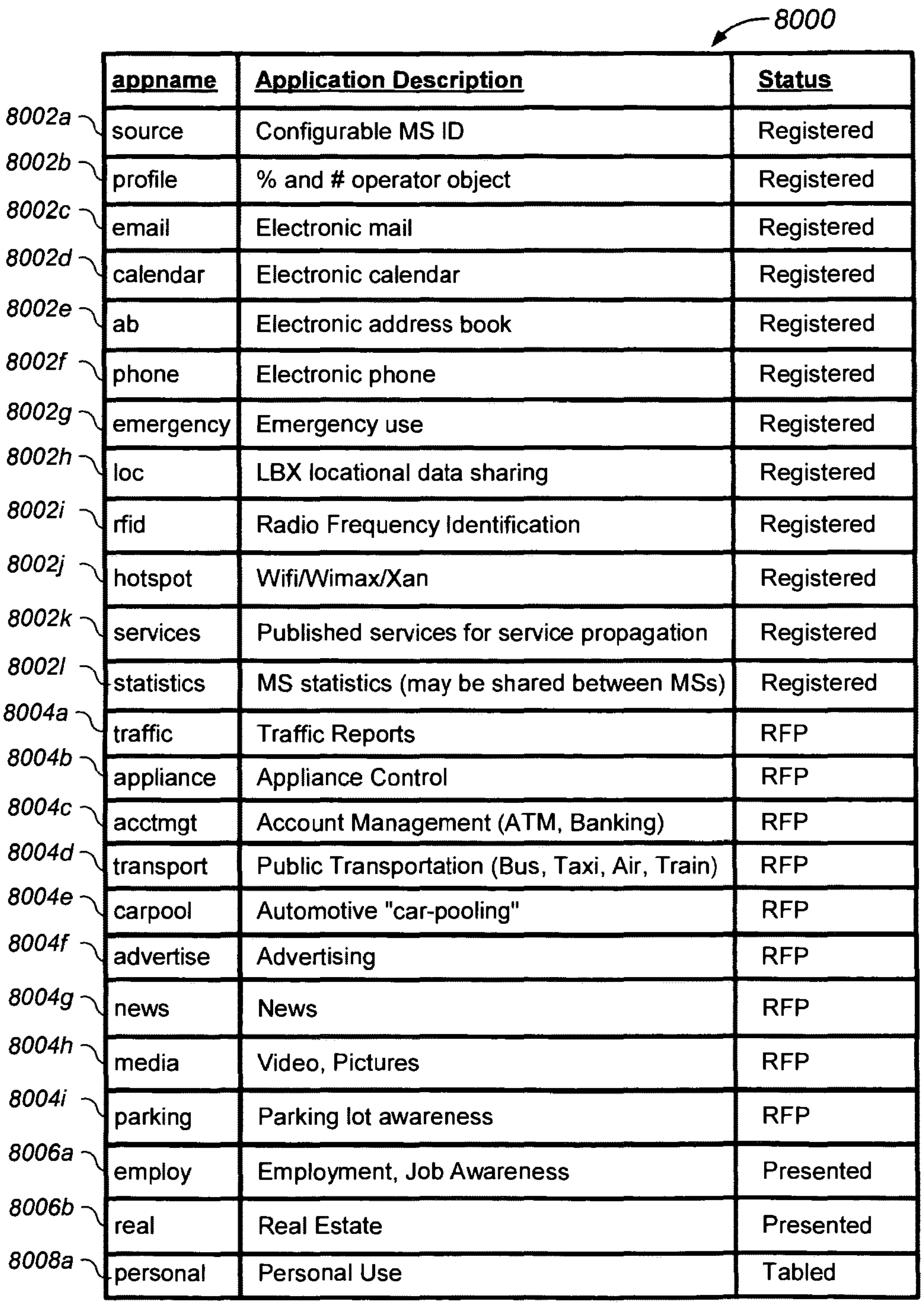

7952
START - Profile match operator evaluation
7954
Access attempt profile; Access charter expression portion; Access reference profile;
7956
Qualified with tag(s)?
Yes
No
7958
TAG_CHECK_LIST = all leaf node tags with data of reference profile
7960
TAG_DATA_MATCH_ATTEMPTS = 0; TAG_DATA_MATCHES = 0;
7962
TAG_CHECK_LIST = specified tag(s) with data of reference profile
7964
Get next tag from TAG_CHECK_LIST
7966
All processed?
Yes
No
7968
Access data for matching tag from attempt profile
7970
Found data?
No
Yes
7972
Increment TAG_DATA_MATCH_ATTEMPTS by # data elements
7974
Get next data element of reference profile tag data
7976
All processed?
Yes
No
7978
Increment TAG_DATA_MATCH_ATTEMPTS by 1
7980
Data element in attempt profile data?
Yes
No
7982
Increment TAG_DATA_MATCHES by 1
7984
Operator = # ?
No
Yes
7986
# condition true?
Yes
No
7992
Calculate Percentage
7994
% condition true?
No
Yes
7988
Return TRUE
7990
Return FALSE

Fig. 79D



	appname	Application Description	Status
8002a	source	Configurable MS ID	Registered
8002b	profile	% and # operator object	Registered
8002c	email	Electronic mail	Registered
8002d	calendar	Electronic calendar	Registered
8002e	ab	Electronic address book	Registered
8002f	phone	Electronic phone	Registered
8002g	emergency	Emergency use	Registered
8002h	loc	LBX locational data sharing	Registered
8002i	rfid	Radio Frequency Identification	Registered
8002j	hotspot	Wifi/Wimax/Xan	Registered
8002k	services	Published services for service propagation	Registered
8002l	statistics	MS statistics (may be shared between MSs)	Registered
8004a	traffic	Traffic Reports	RFP
8004b	appliance	Appliance Control	RFP
8004c	acctmgt	Account Management (ATM, Banking)	RFP
8004d	transport	Public Transportation (Bus, Taxi, Air, Train)	RFP
8004e	carpool	Automotive "car-pooling"	RFP
8004f	advertise	Advertising	RFP
8004g	news	News	RFP
8004h	media	Video, Pictures	RFP
8004i	parking	Parking lot awareness	RFP
8006a	employ	Employment, Job Awareness	Presented
8006b	real	Real Estate	Presented
8008a	personal	Personal Use	Tabled

Fig. 80A

8002a

Field 1100k reference	Description
appfld.source.id.X	MS ID context sensitive default value (X in [email, phone, calendar, ab, rfid, ip (uses first), one for each candidate id for contextual elaboration of an Expression...])
appfld.source.type	MS type
appfld.source.mfr	MS manufacturer
appfld.source.semo	MS serial number
appfld.source.ip	Current delimited (e.g. semicolon) IP address(es) of MS (may have > 1)
...	...

8002b

Field 1100k reference	Description
appfld.profile.contents	MS profile info (e.g.for % or # operator)
...	...

8002c

Field 1100k reference	Description
appfld.email.source	Primary MS email app sender ID
appfld.email.default.X	Email compose defaults wherein X in [attribute.Y such that Y in [cod, urgent, charcode, one for each settable attribute for email...], salutation, doctype, recips, encrypt, compress, one for each email default variable...]
appfld.email.type	Email app type/name
appfld.email.pending.X	Pending email in progress of being composed: X in [see .default.X fields above, cdt, content, one for each email variable...]
appfld.email.last.X.Y.Z	Last email sent (i.e. X = sent) or last email received (i.e. X = rcvd) to/from Y (Y in [ANY, {id} such that id specific source/destination (e.g. joe@yahoo.com)]; Z in [see .pending.X fields above]
...	...

Fig. 80B-1

8002e

Field 1100k reference	Description
appfld.ab.id	Override for AB identifer.
appfld.ab.default.X	AB entry compose defaults wherein X in [attribute.Y such that Y in [marker, color, font, size, one for each settable attribute for AB entry…], background, one for each AB entry default variable...]
appfld.ab.type	AB app type/name
appfld.ab.pending.X	Pending AB entry in progress of being composed: X in [see .default.X fields above, cdt, content, group, one for each AB entry variable...]
appfld.ab.last.X.Y.Z	Last AB entry created locally (i.e. X = local) or last AB entry received (i.e. X = other) to/from Y (Y in [ANY, {id} such that id specific source/destination (e.g. MSID4F3EB2398)]; Z in [see .pending.X fields above]
…	…

8002d

Field 1100k reference	Description
appfld.calendar.id	Calendar id override.
appfld.calendar.next.X	MS calendar next entry data X
appfld.calendar.nextavail.X	MS calendar app next available free slots
appfld.calendar.default.X	Calendar entry compose defaults wherein X in [attribute.Y such that Y in [cod, urgent, color, one for each settable attribute for calendar entry…], recips, camp, one for each calendar entry default variable...]
appfld.calendar.sched.X	MS calendar schedule for period X (X in [curweek, curmonth, curyear, etc])
appfld.calendar.type	MS calendar app type/name
appfld.calendar.pending.X	Pending calendar entry in progress of being composed: X in [see .default.X fields above, cdt, content, date/time(s), recurring, one for each calendar entry variable...]
appfld.calendar.last.X.Y.Z	Last calendar entry created locally (i.e. X = local) or last calendar entry received (i.e. X = other) to/from Y (Y in [ANY, {id} such that id specific source/destination (e.g. joe@yahoo.com)]; Z in [see .pending.X fields above]
…	

Fig. 80B-2

8002f

Field 1100k reference	Description
appfld.phone.id	MS phone app primary caller id
appfld.phone.default.X	Phone call defaults wherein X in [volume, encrypt, compress, camp, one for each phone application default variable...]
appfld.phone.caller	MS phone app caller id override
appfld.phone.log.X	MS phone app log file X (X in [out, in, missed, one for each log…])
appfld.phone.record.X	MS phone app record boolean for calls made, calls received, or specific numbers (e.g. MS IDs)
appfld.phone.ogm	MS phone app OGM
appfld.phone.dt.X	Date/time stamp for X (X in [tx (last call made), rx (last call received), missed (last call missed)])
appfld.phone.type	MS phone app type/name
appfld.phone.fwd	MS phone app forwarding setting for prioritized list. A prioritized list automatically tries the next entry if there is no answer or a failed outbound connection.
appfld.phone.ring	Ring setting = ring tone selection reference OR audio file reference.
appfld.phone.vibe	Vibration setting = None OR reference for vibration type.
appfld.phone.droplocs	MS phone dropped locations
appfld.phone.macro.X	Automated macros for ARU interfaces
appfld.phone.pwd.X	MS phone passwords for allowing calls to complete and for variable processing by caller.
appfld.phone.msg.X	Phone messages (new, saved, etc)
appfld.phone.blackout	MS phone blackout conditions (expressions including application in use, date/time(s), current location(s), any MS detectable condition)
appfld.phone.pending.X	Pending phone call: X in [see .default.X fields above, cdt, data, one for each phone call app variable...]; data is only present for peer to peer MS phone calls (it carries the voice call).
appfld.phone.last.X.Y.Z	Last phone call made (i.e. X = out) or last call received (i.e. X = in) to/from Y (Y in [ANY, {id} such that id specific source/destination (e.g. MSID4F3EB2398)]; Z in [see .pending.X fields above, edt]
…	

Fig. 80B-3

8002g

Field 1100k reference	**Description**
appfld.emergency.type	Emergency type (Police, Fire, Amber, Help, Caution, etc)
appfld.emergency.cdt	Emergency create date/time stamp
appfld.emergency.duration	Emergency anticipated duration
appfld.emergency.content.X	Emergency content.X (X in [type, alert, prefmeth, one field for each alert content section…])
appfld.emergency.method.X	MS emergency notify method X (X in [attribute.Y, how, where])
appfld.emergency.last.X	Last emergency WDR data: X in [self, other].
…	…

8002h

Field 1100k reference	**Description**
appfld.loc.blackout	Blackout criteria
appfld.loc.mode	Current MS mode
appfld.loc.geofence.X	MS configured geofence data
appfld.loc.halo.X	MS configured interest perimeter
appfld.loc.mark.X	X = # of saved location marks Y
appfld.loc.dcdb.X	Location activated delivery content for MS ID X
appfld.loc.beacon.X	Beacon a peer MS at the peer MS; Sending MS controls what is seen by receiving MS
…	…

8002i

Field 1100k reference	**Description**
appfld.rfid.id	Deaults appfld.source.id.rfid.
appfld.rfid.passive.X	MS Passive RFI capability
appfld.rfid.active.X	MS Active RFI capability
appfld.rfid.listen.X	Active RFID listening channel directions
appfld.rfid.seek.X	Passive RFID polling channel directions
…	…

8002j

Field 1100k reference	**Description**
appfld.hotspot.listen	Listening boolean
appfld.hotspot.X	X number of Hotspot(s) (the information) automatically detected over time
…	…

8002k

Field 1100k reference	**Description**
appfld.services.X	LN-expanse dynamic services information
…	…

Fig. 80B-4

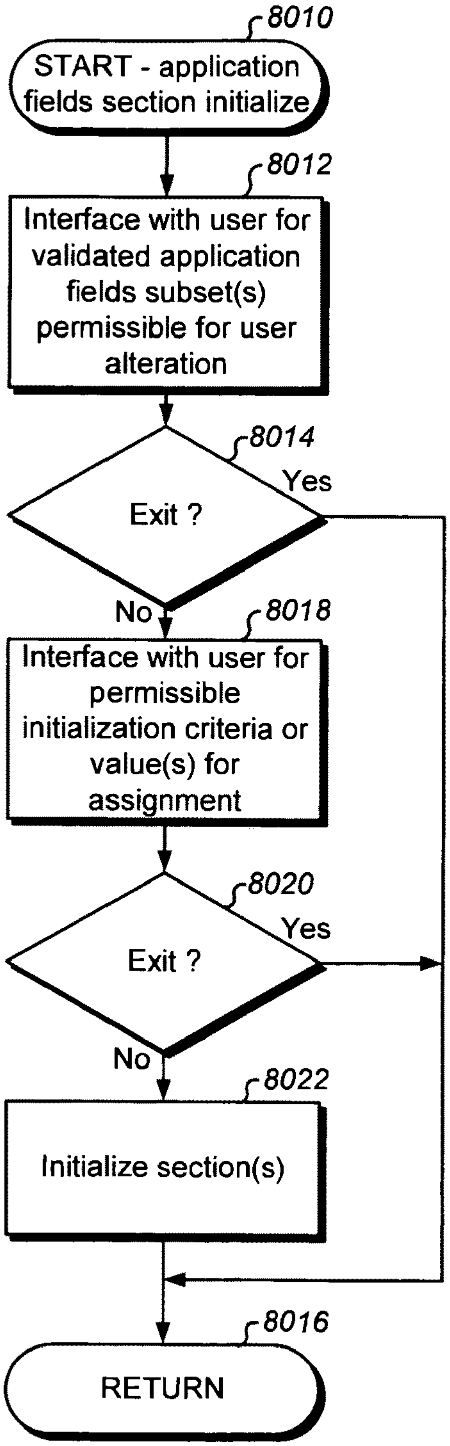

8010
START - application fields section initialize
8012
Interface with user for validated application fields subset(s) permissible for user alteration
8014
Exit ?
Yes
No
8018
Interface with user for permissible initialization criteria or value(s) for assignment
8020
Exit ?
Yes
No
8022
Initialize section(s)
8016
RETURN

Fig. 80C

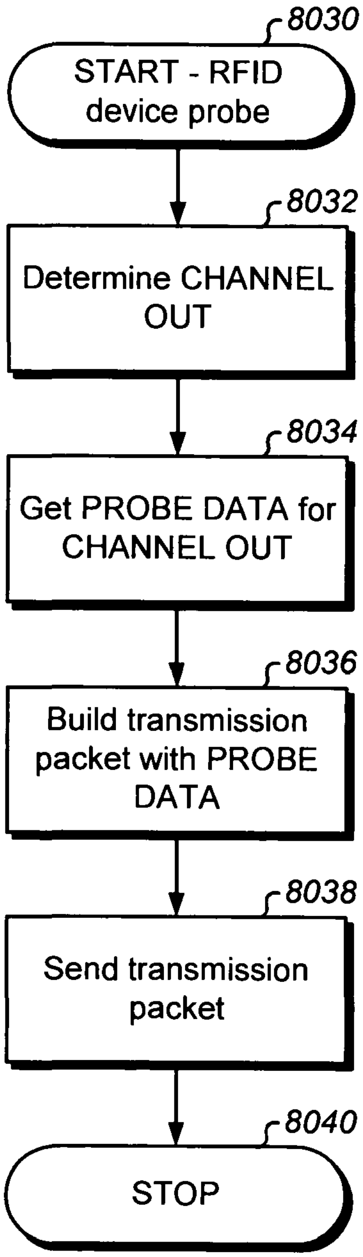

8030
START - RFID device probe
8032
Determine CHANNEL OUT
8034
Get PROBE DATA for CHANNEL OUT
8036
Build transmission packet with PROBE DATA
8038
Send transmission packet
8040
STOP

Fig. 80D

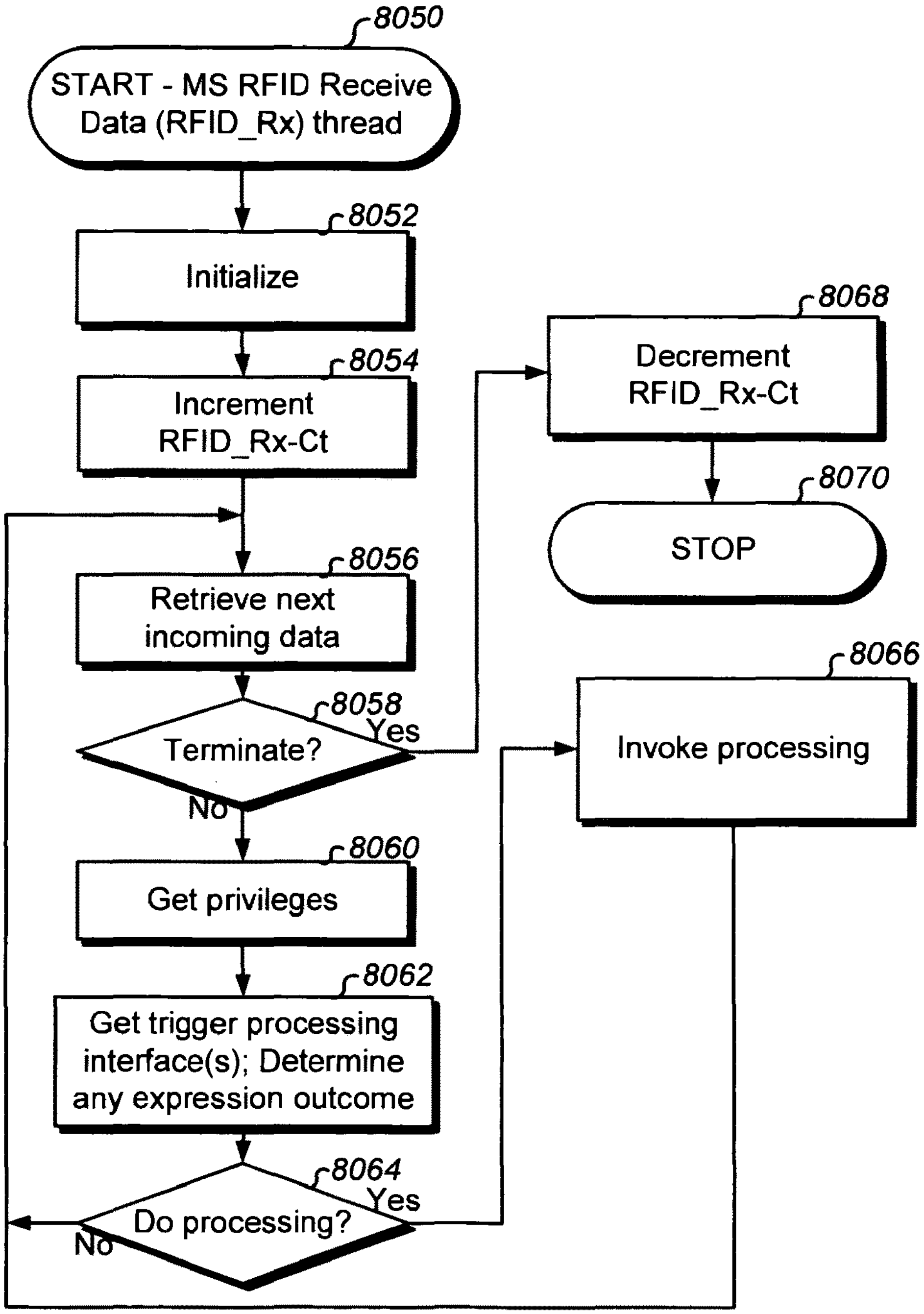

8050
START - MS RFID Receive Data (RFID_Rx) thread
8052
Initialize
8054
Increment RFID_Rx-Ct
8056
Retrieve next incoming data
8058
Terminate?
Yes
No
8060
Get privileges
8062
Get trigger processing interface(s); Determine any expression outcome
8064
Do processing?
Yes
No
8066
Invoke processing
8068
Decrement RFID_Rx-Ct
8070
STOP

Fig. 80E

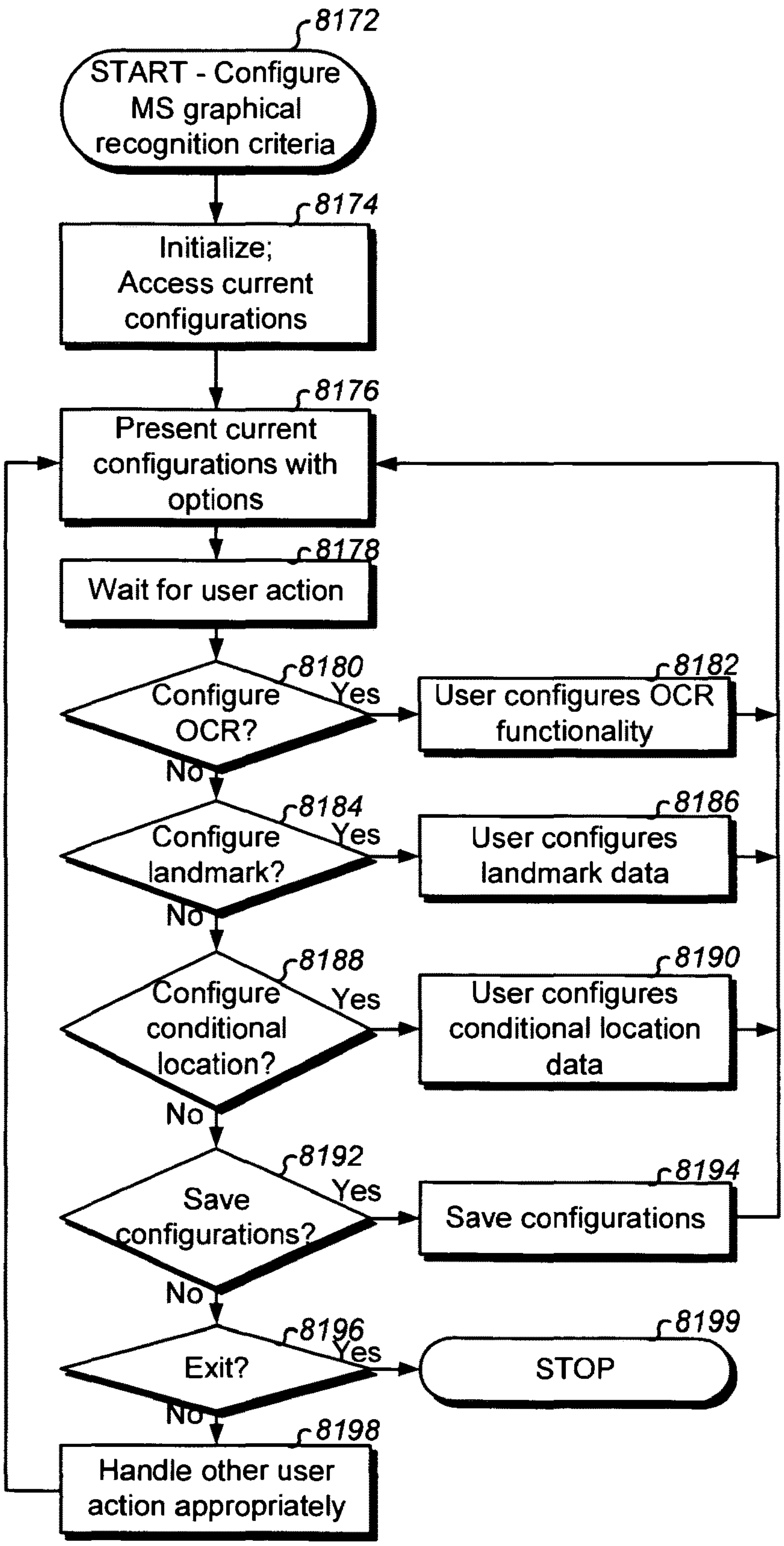

8172
START - Configure MS graphical recognition criteria
8174
Initialize; Access current configurations
8176
Present current configurations with options
8178
Wait for user action
8180
Configure OCR?
Yes
No
8182
User configures OCR functionality
8184
Configure landmark?
Yes
No
8186
User configures landmark data
8188
Configure conditional location?
Yes
No
8190
User configures conditional location data
8192
Save configurations?
Yes
No
8194
Save configurations
8196
Exit?
Yes
No
8199
STOP
8198
Handle other user action appropriately

Fig. 81A

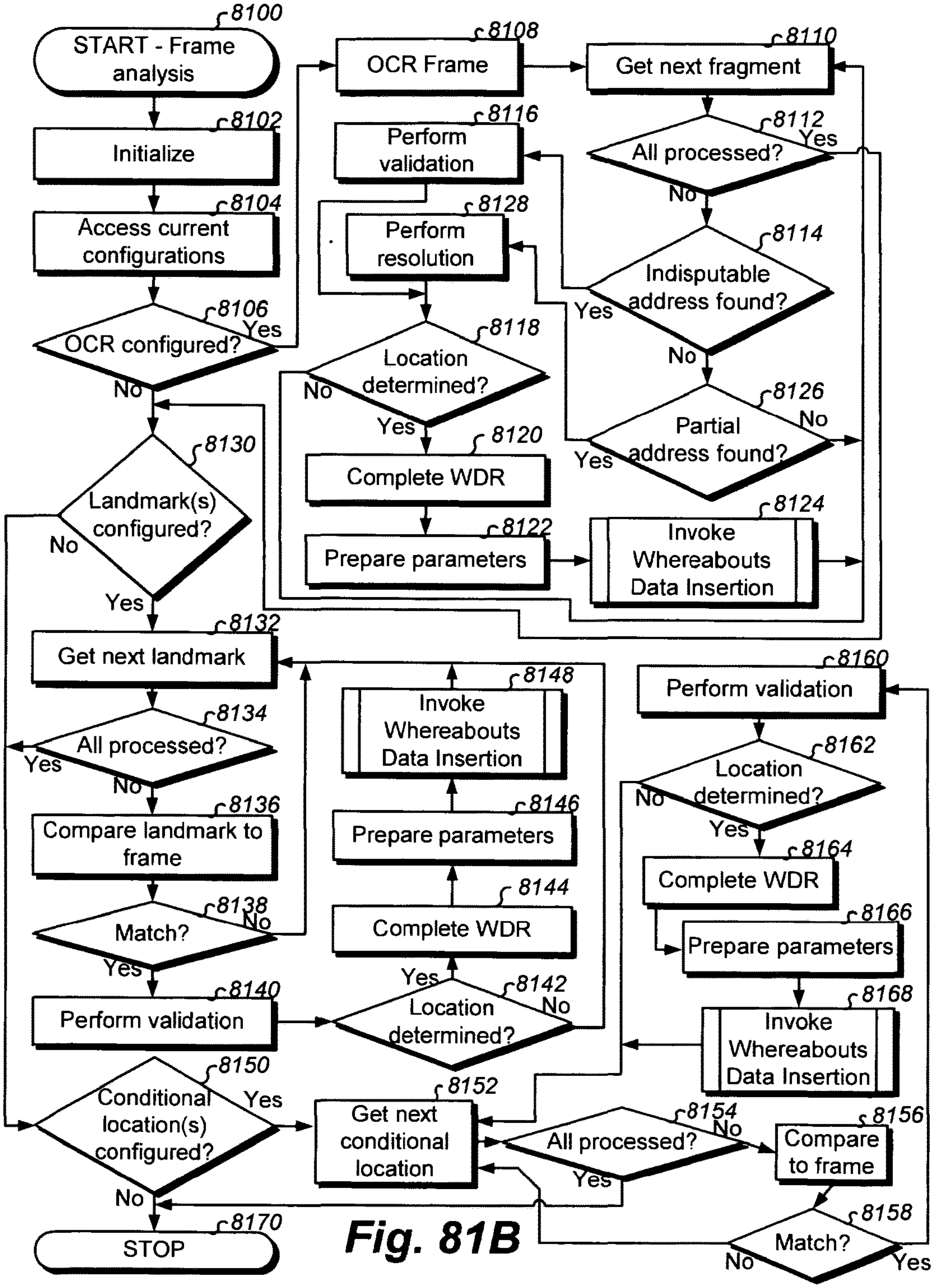

8100
START - Frame analysis
8102
Initialize
8104
Access current configurations
8106
OCR configured?
Yes
No
8108
OCR Frame
8110
Get next fragment
8112
All processed?
Yes
No
8114
Indisputable address found?
Yes
No
8116
Perform validation
8128
Perform resolution
8118
Location determined?
No
Yes
8120
Complete WDR
8122
Prepare parameters
8124
Invoke Whereabouts Data Insertion
8126
Partial address found?
Yes
No
8130
Landmark(s) configured?
No
Yes
8132
Get next landmark
8134
All processed?
Yes
No
8136
Compare landmark to frame
8138
Match?
No
Yes
8140
Perform validation
8142
Location determined?
Yes
No
8144
Complete WDR
8146
Prepare parameters
8148
Invoke Whereabouts Data Insertion
8150
Conditional location(s) configured?
Yes
No
8152
Get next conditional location
8154
All processed?
No
Yes
8156
Compare to frame
8158
Match?
No
Yes
8160
Perform validation
8162
Location determined?
No
Yes
8164
Complete WDR
8166
Prepare parameters
8168
Invoke Whereabouts Data Insertion
8170
STOP

Fig. 81B

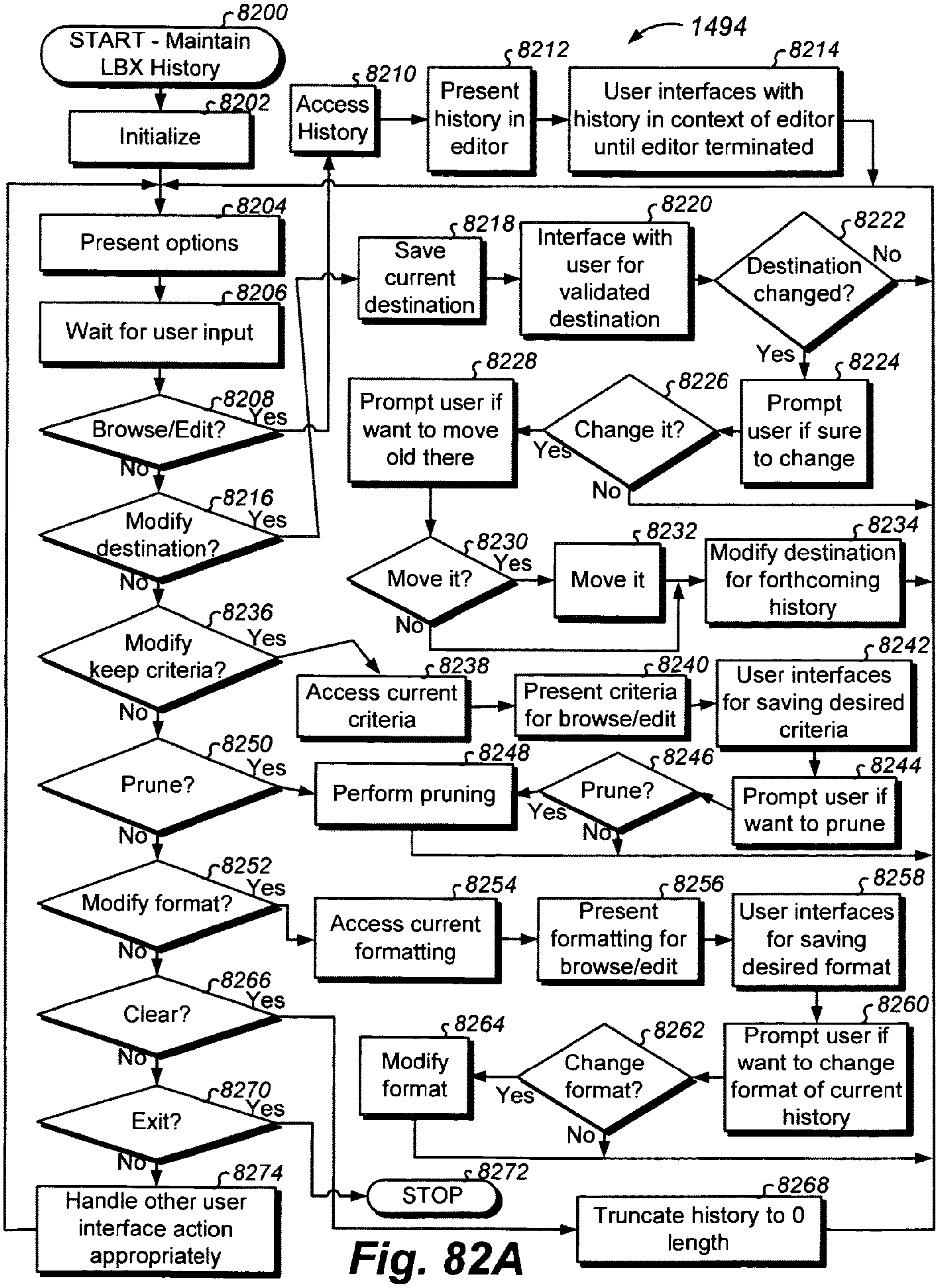

1494
8200
START - Maintain LBX History
8202
Initialize
8210
Access History
8212
Present history in editor
8214
User interfaces with history in context of editor until editor terminated
8204
Present options
8206
Wait for user input
8208
Browse/Edit?
Yes
No
8216
Modify destination?
Yes
No
8218
Save current destination
8220
Interface with user for validated destination
8222
Destination changed?
No
Yes
8224
Prompt user if sure to change
8226
Change it?
Yes
No
8228
Prompt user if want to move old there
8230
Move it?
Yes
No
8232
Move it
8234
Modify destination for forthcoming history
8236
Modify keep criteria?
Yes
No
8238
Access current criteria
8240
Present criteria for browse/edit
8242
User interfaces for saving desired criteria
8250
Prune?
Yes
No
8248
Perform pruning
8246
Prune?
Yes
No
8244
Prompt user if want to prune
8252
Modify format?
Yes
No
8254
Access current formatting
8256
Present formatting for browse/edit
8258
User interfaces for saving desired format
8266
Clear?
Yes
No
8260
Prompt user if want to change format of current history
8262
Change format?
Yes
No
8264
Modify format
8270
Exit?
Yes
No
8272
STOP
8274
Handle other user interface action appropriately
8268
Truncate history to 0 length

Fig. 82A

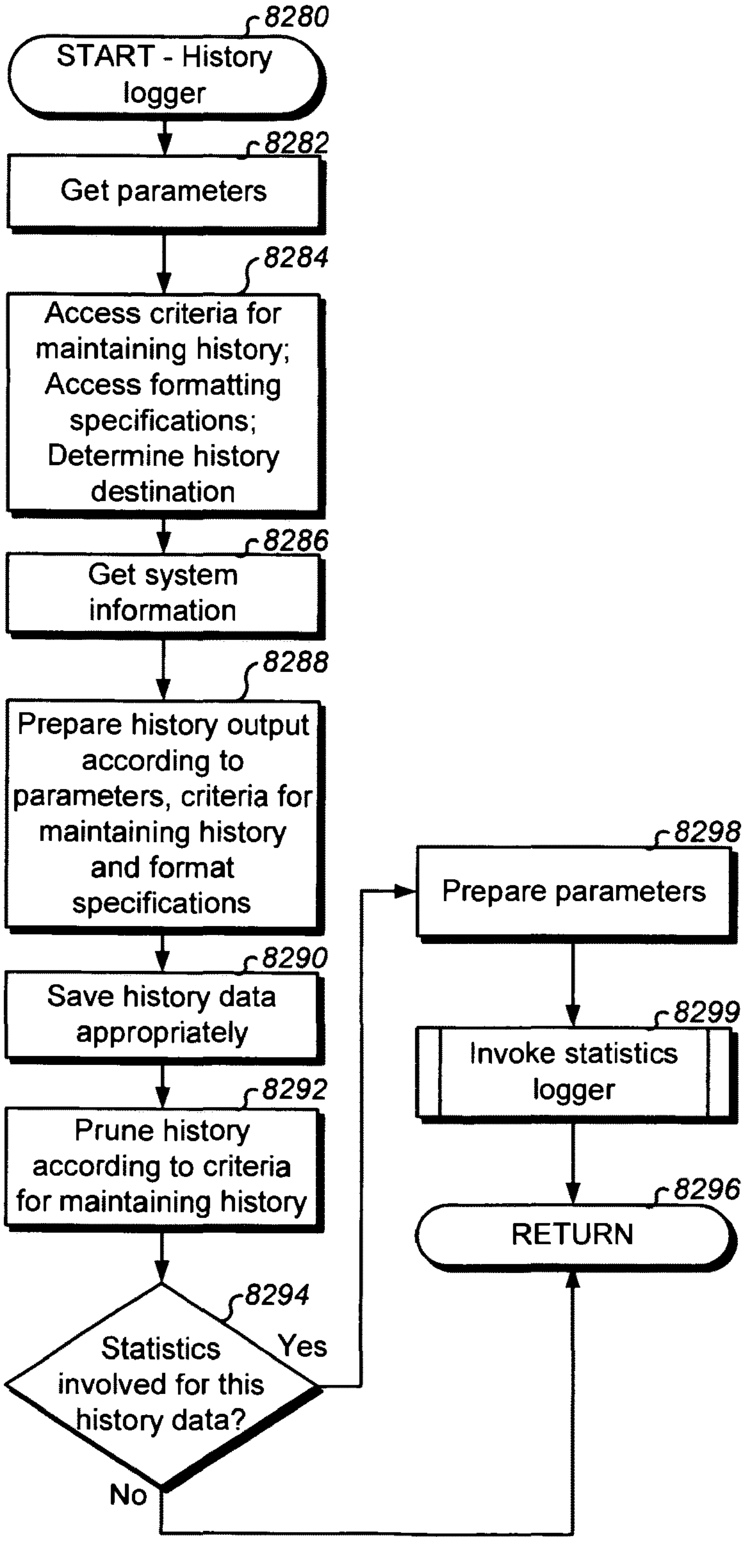

8280
START - History logger
8282
Get parameters
8284
Access criteria for maintaining history; Access formatting specifications; Determine history destination
8286
Get system information
8288
Prepare history output according to parameters, criteria for maintaining history and format specifications
8290
Save history data appropriately
8292
Prune history according to criteria for maintaining history
8294
Statistics involved for this history data?
Yes
No
8298
Prepare parameters
8299
Invoke statistics logger
8296
RETURN

Fig. 82B

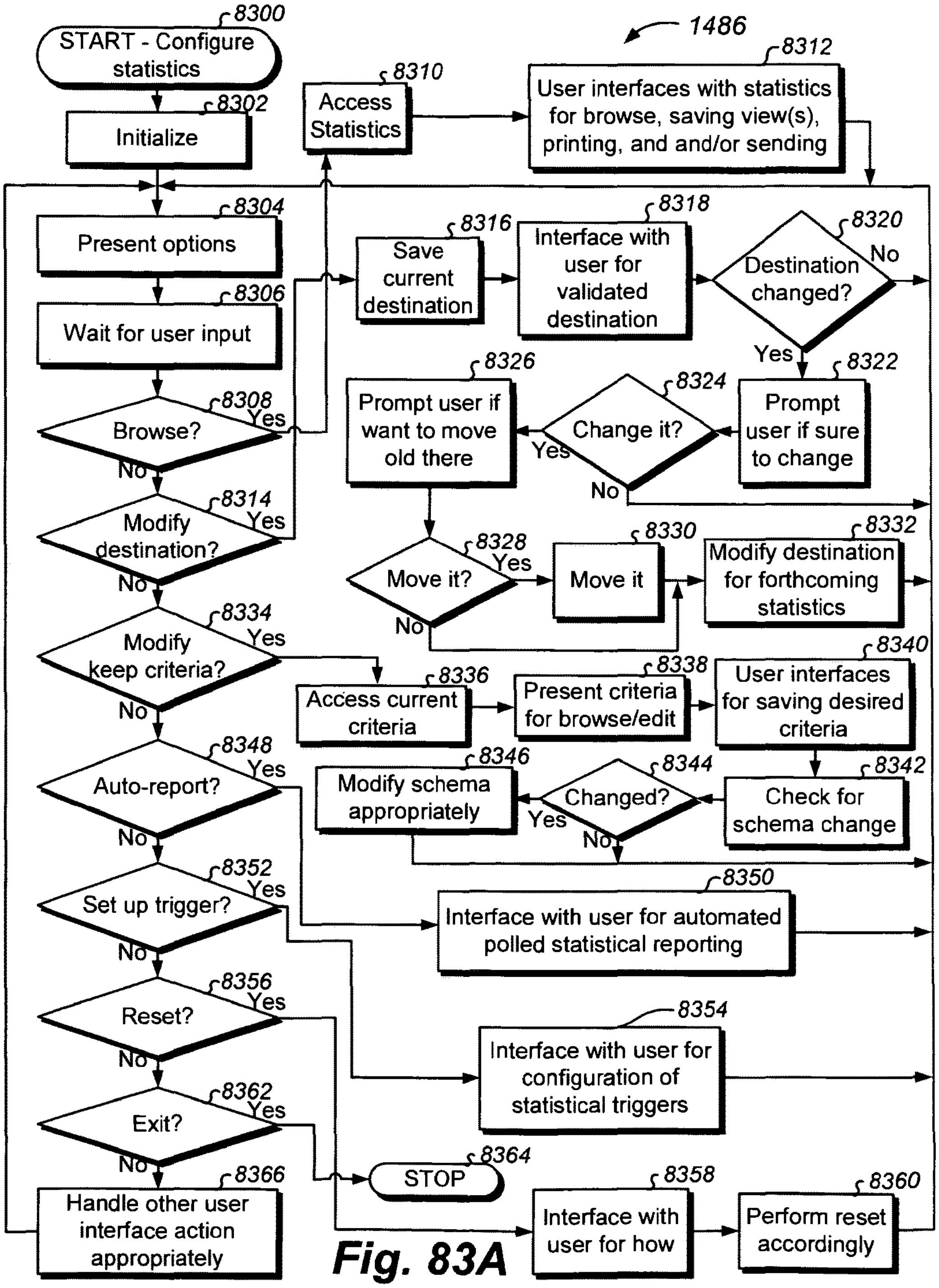

1486
8300 START - Configure statistics
8302 Initialize
8304 Present options
8306 Wait for user input
8308 Browse?
Yes
No
8310 Access Statistics
8312 User interfaces with statistics for browse, saving view(s), printing, and and/or sending
8314 Modify destination?
Yes
No
8316 Save current destination
8318 Interface with user for validated destination
8320 Destination changed?
No
Yes
8322 Prompt user if sure to change
8324 Change it?
Yes
No
8326 Prompt user if want to move old there
8328 Move it?
Yes
No
8330 Move it
8332 Modify destination for forthcoming statistics
8334 Modify keep criteria?
Yes
No
8336 Access current criteria
8338 Present criteria for browse/edit
8340 User interfaces for saving desired criteria
8342 Check for schema change
8344 Changed?
Yes
No
8346 Modify schema appropriately
8348 Auto-report?
Yes
No
8350 Interface with user for automated polled statistical reporting
8352 Set up trigger?
Yes
No
8354 Interface with user for configuration of statistical triggers
8356 Reset?
Yes
No
8358 Interface with user for how
8360 Perform reset accordingly
8362 Exit?
Yes
No
8364 STOP
8366 Handle other user interface action appropriately

Fig. 83A

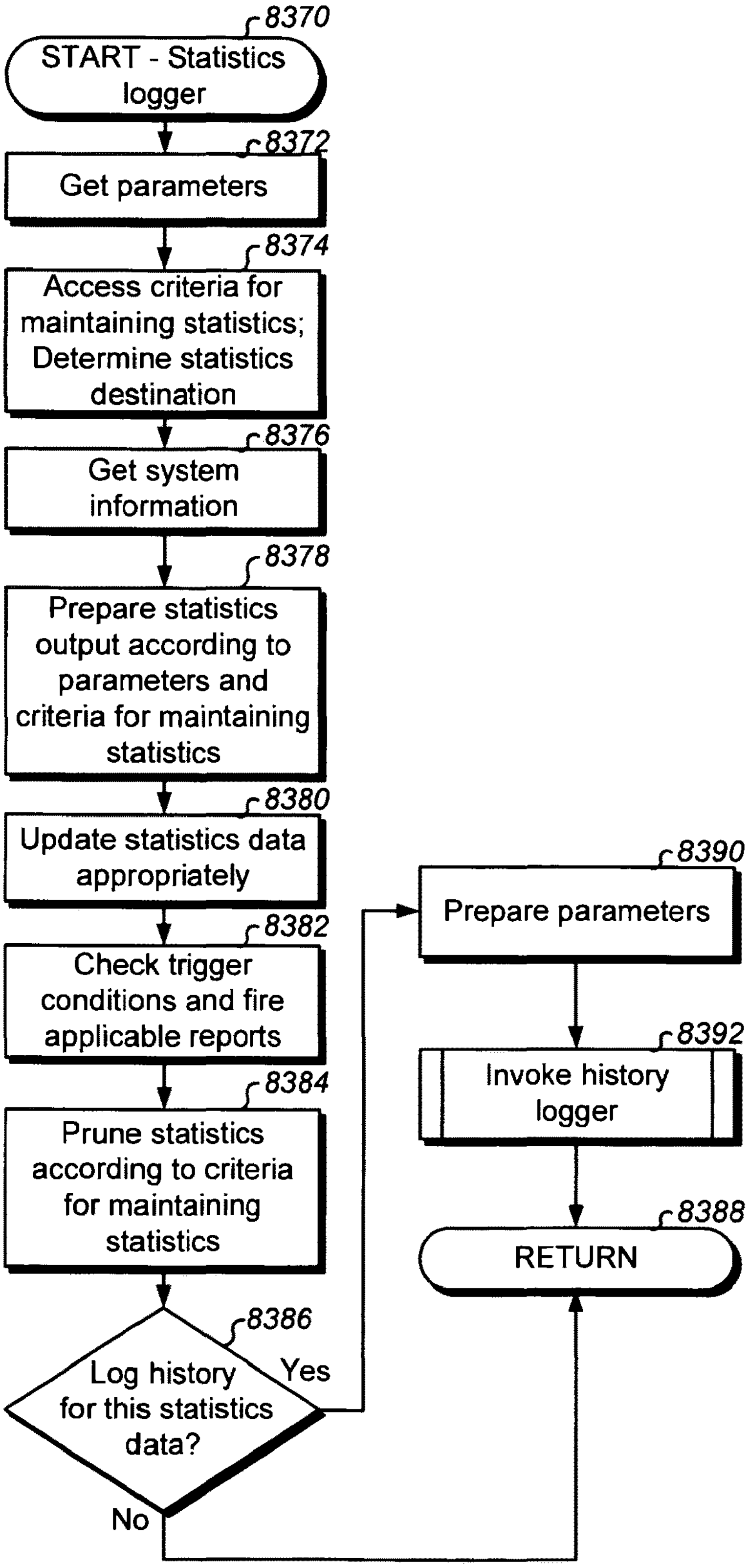

8370
START - Statistics logger
8372
Get parameters
8374
Access criteria for maintaining statistics; Determine statistics destination
8376
Get system information
8378
Prepare statistics output according to parameters and criteria for maintaining statistics
8380
Update statistics data appropriately
8382
Check trigger conditions and fire applicable reports
8384
Prune statistics according to criteria for maintaining statistics
8386
Log history for this statistics data?
Yes
No
8390
Prepare parameters
8392
Invoke history logger
8388
RETURN

Fig. 83B

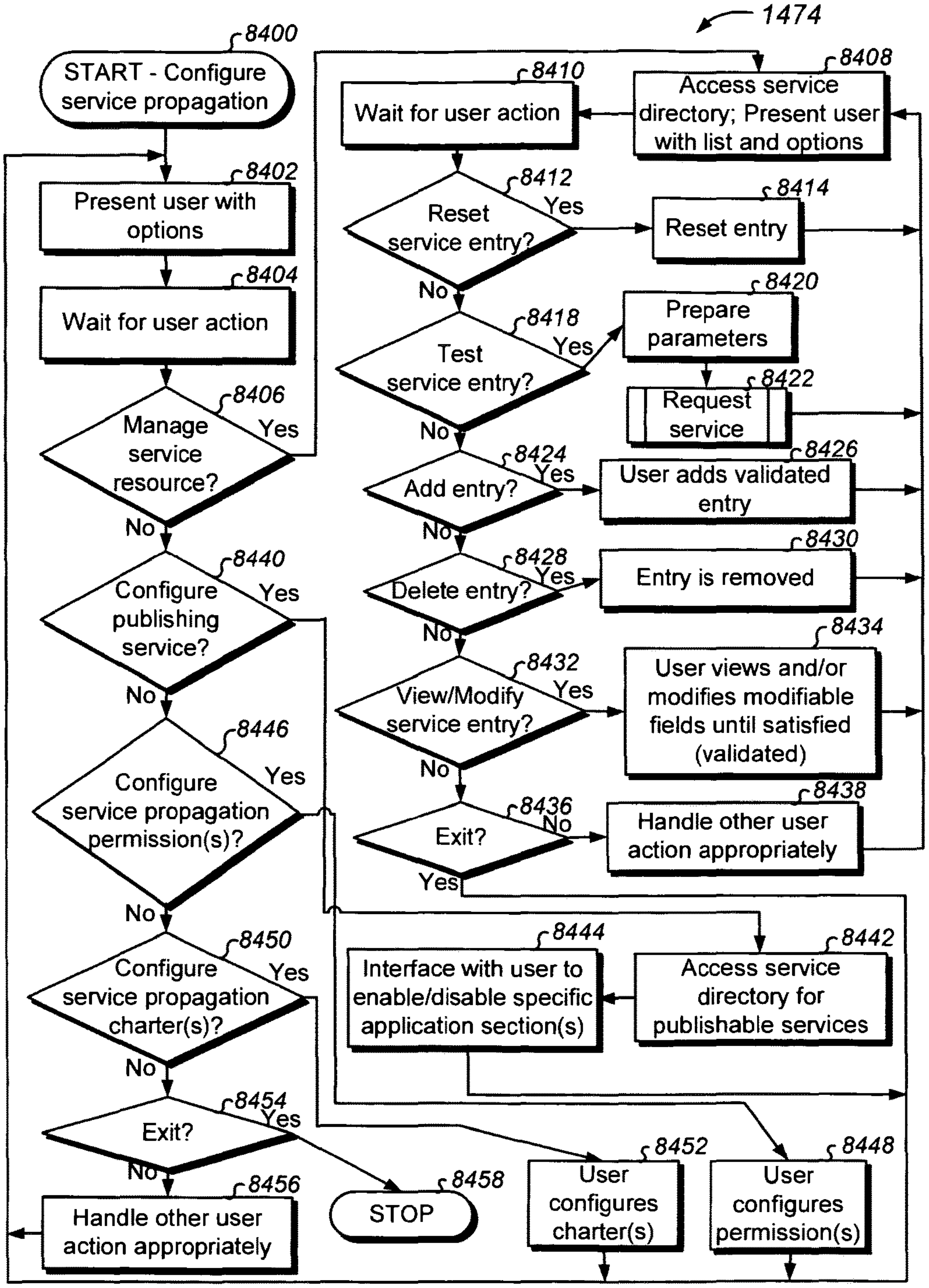

1474
8400
START - Configure service propagation
8402
Present user with options
8404
Wait for user action
8406
Manage service resource?
Yes
No
8440
Configure publishing service?
Yes
No
8446
Configure service propagation permission(s)?
Yes
No
8450
Configure service propagation charter(s)?
Yes
No
8454
Exit?
Yes
No
8456
Handle other user action appropriately
8458
STOP
8410
Wait for user action
8408
Access service directory; Present user with list and options
8412
Reset service entry?
Yes
No
8414
Reset entry
8418
Test service entry?
Yes
No
8420
Prepare parameters
8422
Request service
8424
Add entry?
Yes
No
8426
User adds validated entry
8428
Delete entry?
Yes
No
8430
Entry is removed
8432
View/Modify service entry?
Yes
No
8434
User views and/or modifies modifiable fields until satisfied (validated)
8436
Exit?
No
Yes
8438
Handle other user action appropriately
8444
Interface with user to enable/disable specific application section(s)
8442
Access service directory for publishable services
8452
User configures charter(s)
8448
User configures permission(s)

Fig. 84A

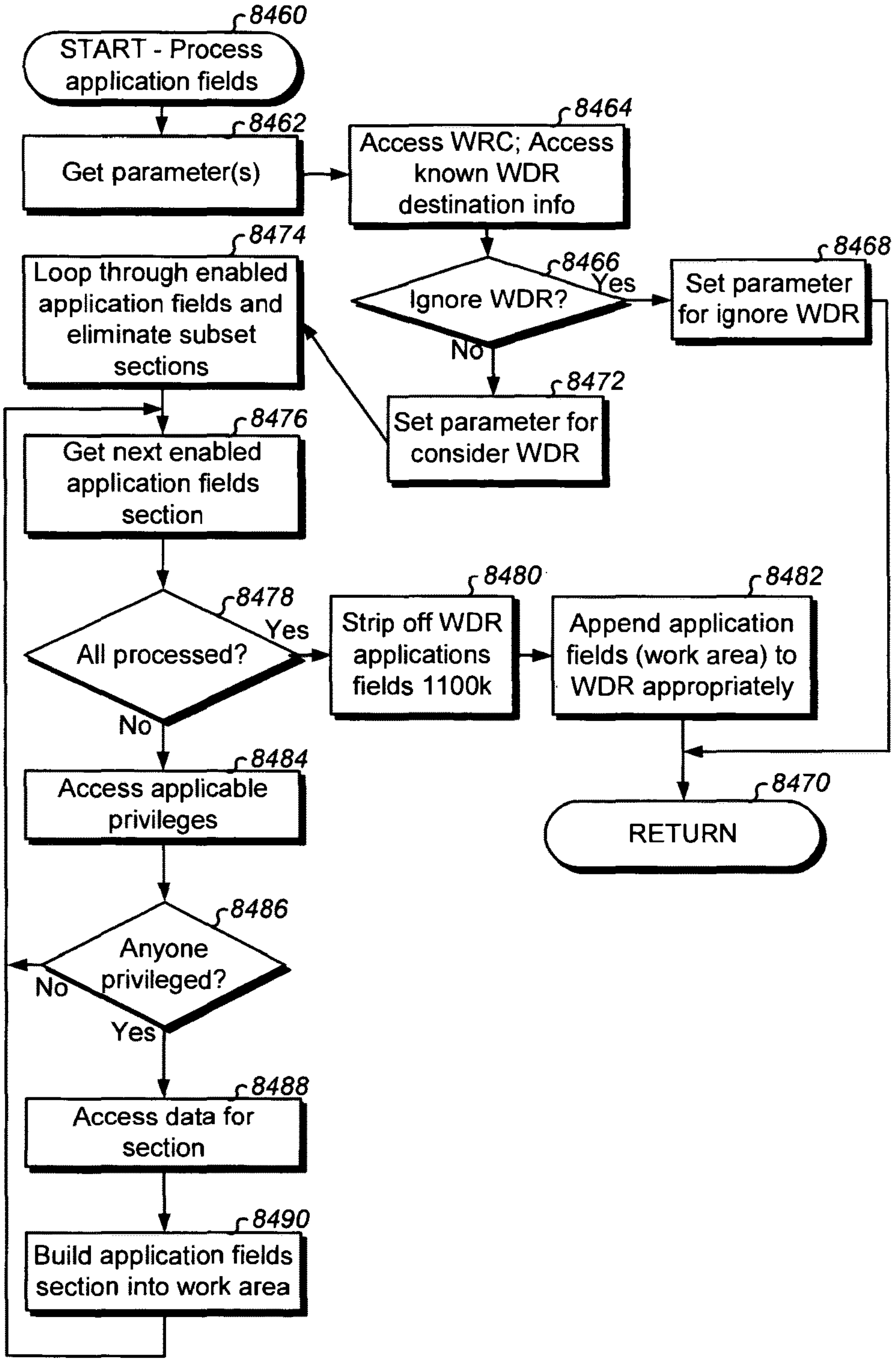

8460
START - Process application fields
8462
Get parameter(s)
8464
Access WRC; Access known WDR destination info
8466
Ignore WDR?
Yes
No
8468
Set parameter for ignore WDR
8472
Set parameter for consider WDR
8474
Loop through enabled application fields and eliminate subset sections
8476
Get next enabled application fields section
8478
All processed?
Yes
No
8480
Strip off WDR applications fields 1100k
8482
Append application fields (work area) to WDR appropriately
8470
RETURN
8484
Access applicable privileges
8486
Anyone privileged?
No
Yes
8488
Access data for section
8490
Build application fields section into work area

Fig. 84B

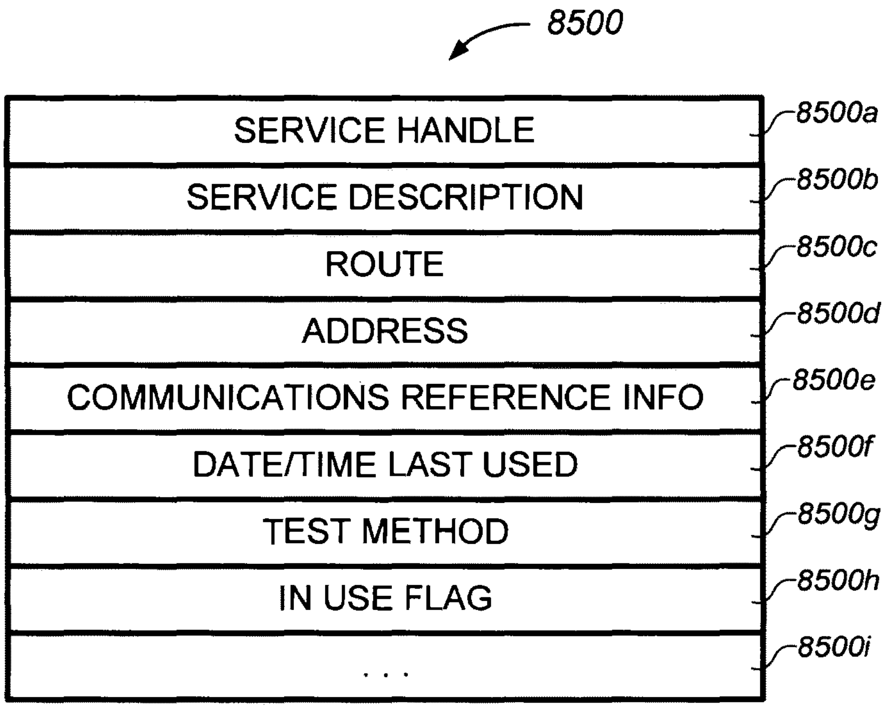

8500
SERVICE HANDLE
8500a
SERVICE DESCRIPTION
8500b
ROUTE
8500c
ADDRESS
8500d
COMMUNICATIONS REFERENCE INFO
8500e
DATE/TIME LAST USED
8500f
TEST METHOD
8500g
IN USE FLAG
8500h
. . .
8500i

Fig. 85A

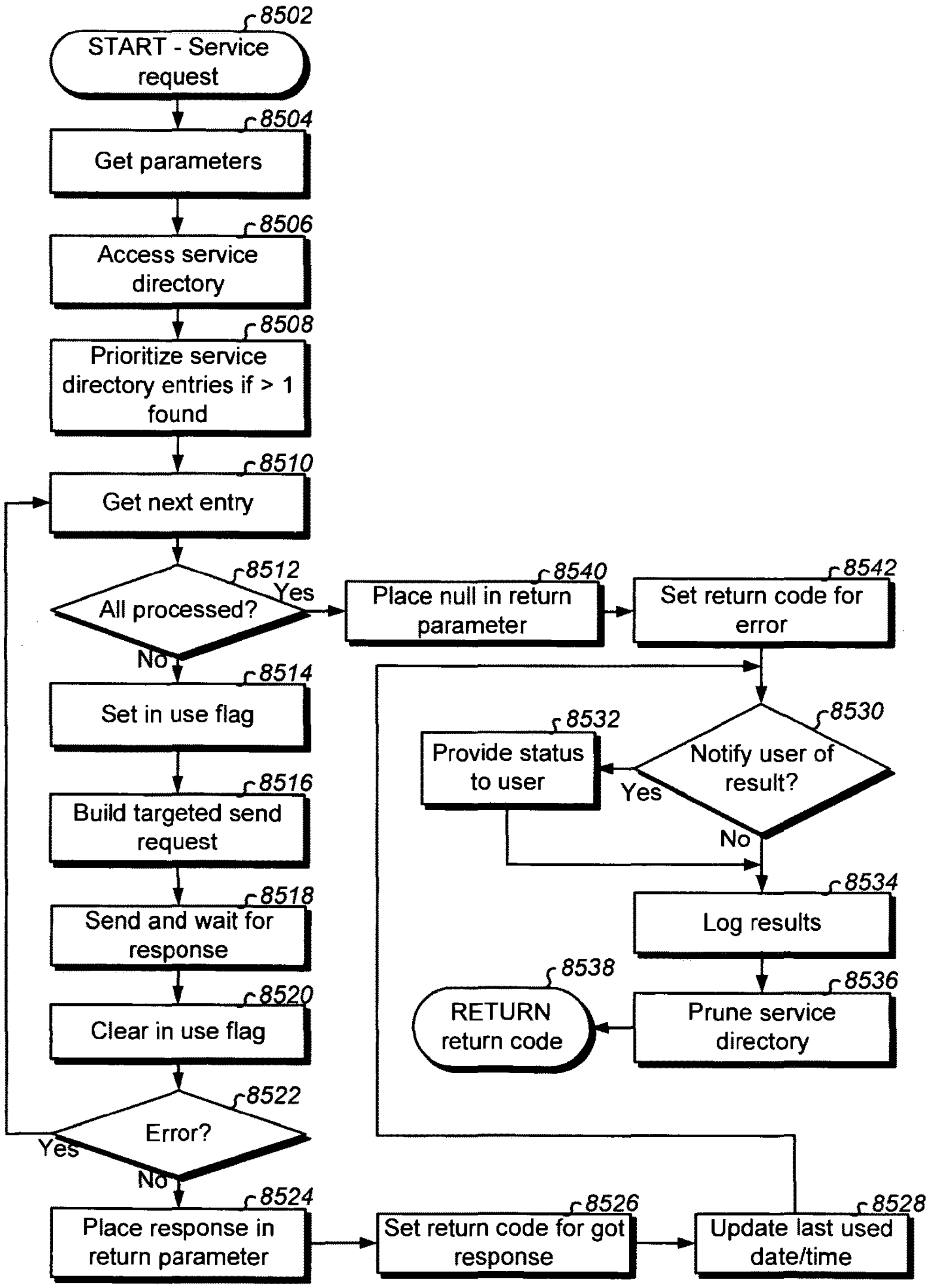

8502
START - Service request
8504
Get parameters
8506
Access service directory
8508
Prioritize service directory entries if > 1 found
8510
Get next entry
8512
All processed?
Yes
No
8514
Set in use flag
8516
Build targeted send request
8518
Send and wait for response
8520
Clear in use flag
8522
Error?
Yes
No
8524
Place response in return parameter
8526
Set return code for got response
8528
Update last used date/time
8540
Place null in return parameter
8542
Set return code for error
8530
Notify user of result?
Yes
No
8532
Provide status to user
8534
Log results
8536
Prune service directory
8538
RETURN return code

Fig. 85B

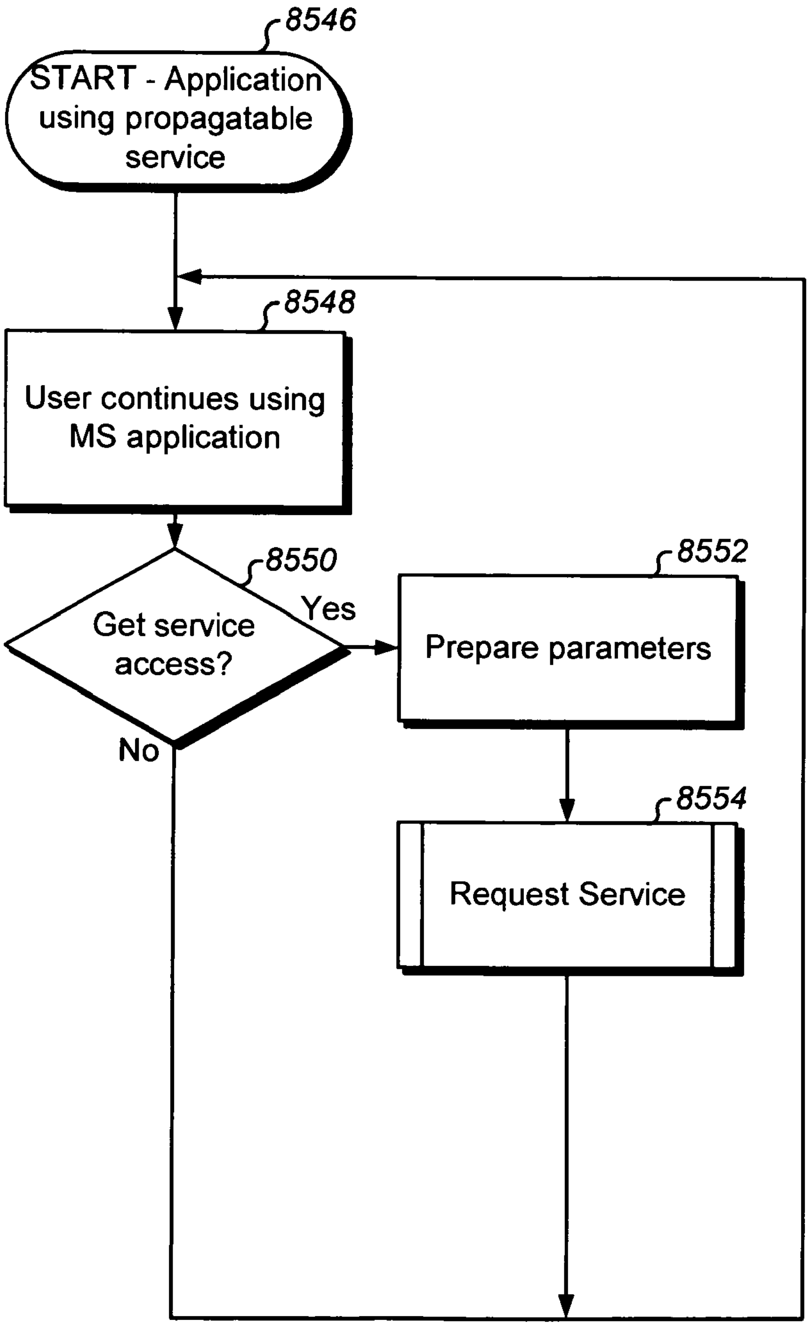

8546
START - Application using propagatable service
8548
User continues using MS application
8550
Get service access?
Yes
No
8552
Prepare parameters
8554
Request Service

Fig. 85C

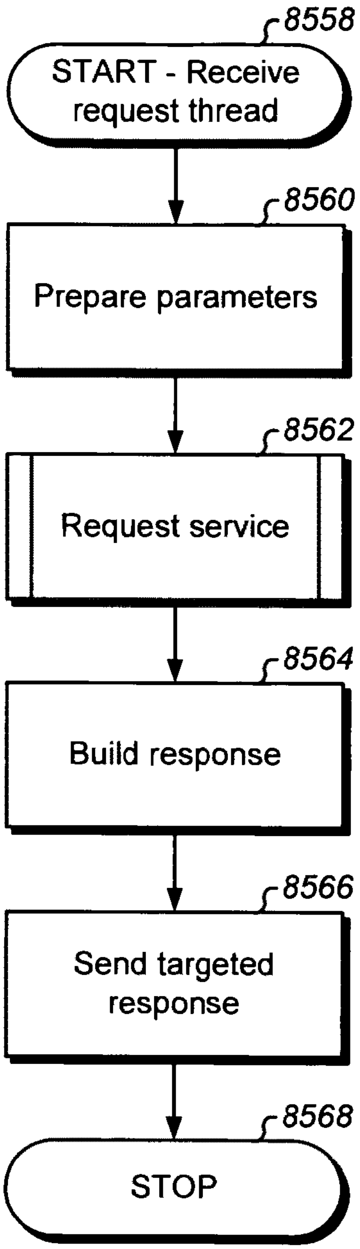

8558
START - Receive request thread
8560
Prepare parameters
8562
Request service
8564
Build response
8566
Send targeted response
8568
STOP

Fig. 85D

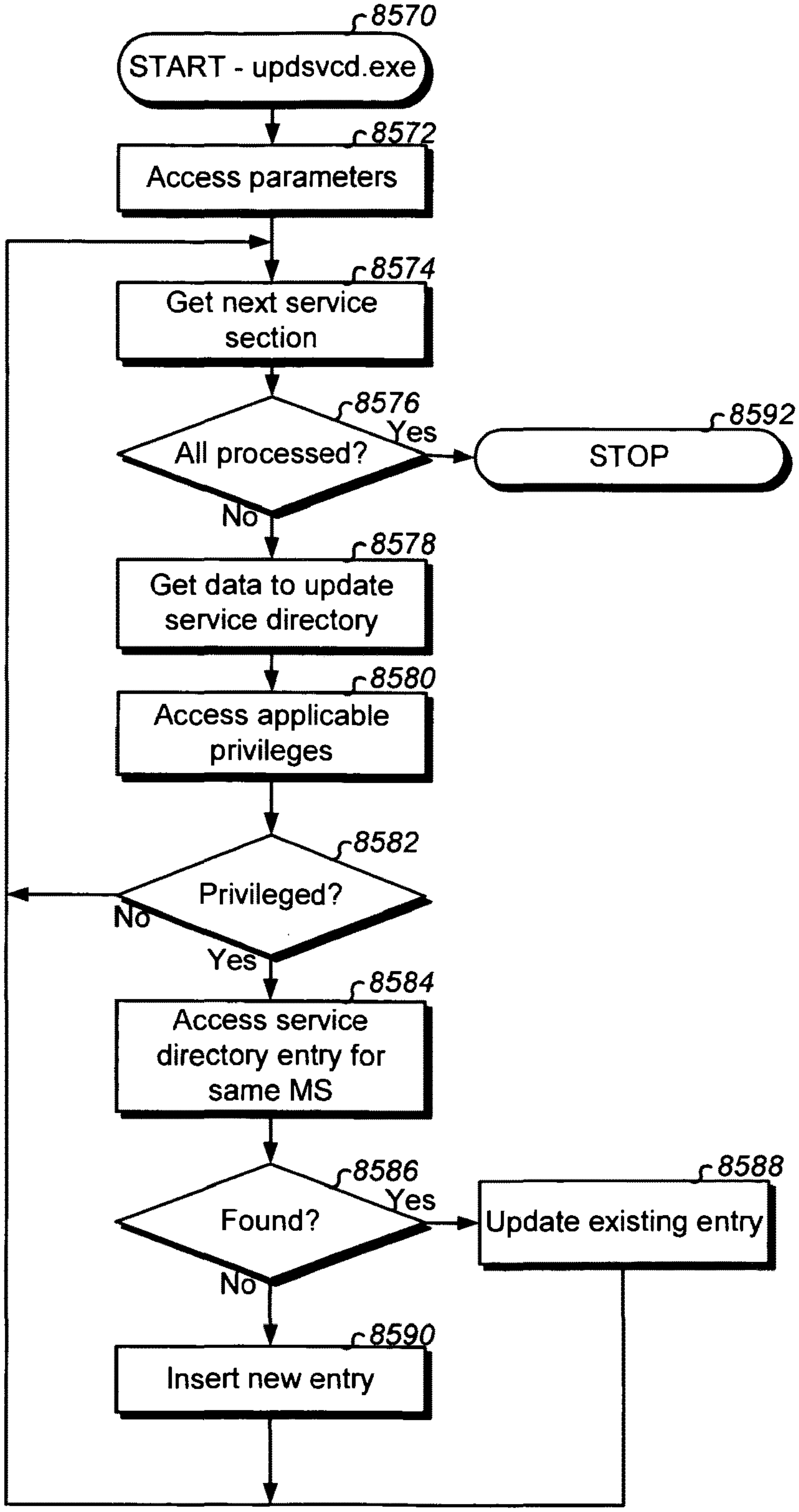

8570
START - updsvcd.exe
8572
Access parameters
8574
Get next service section
8576
All processed?
Yes
8592
STOP
No
8578
Get data to update service directory
8580
Access applicable privileges
8582
Privileged?
No
Yes
8584
Access service directory entry for same MS
8586
Found?
Yes
8588
Update existing entry
No
8590
Insert new entry

Fig. 85E

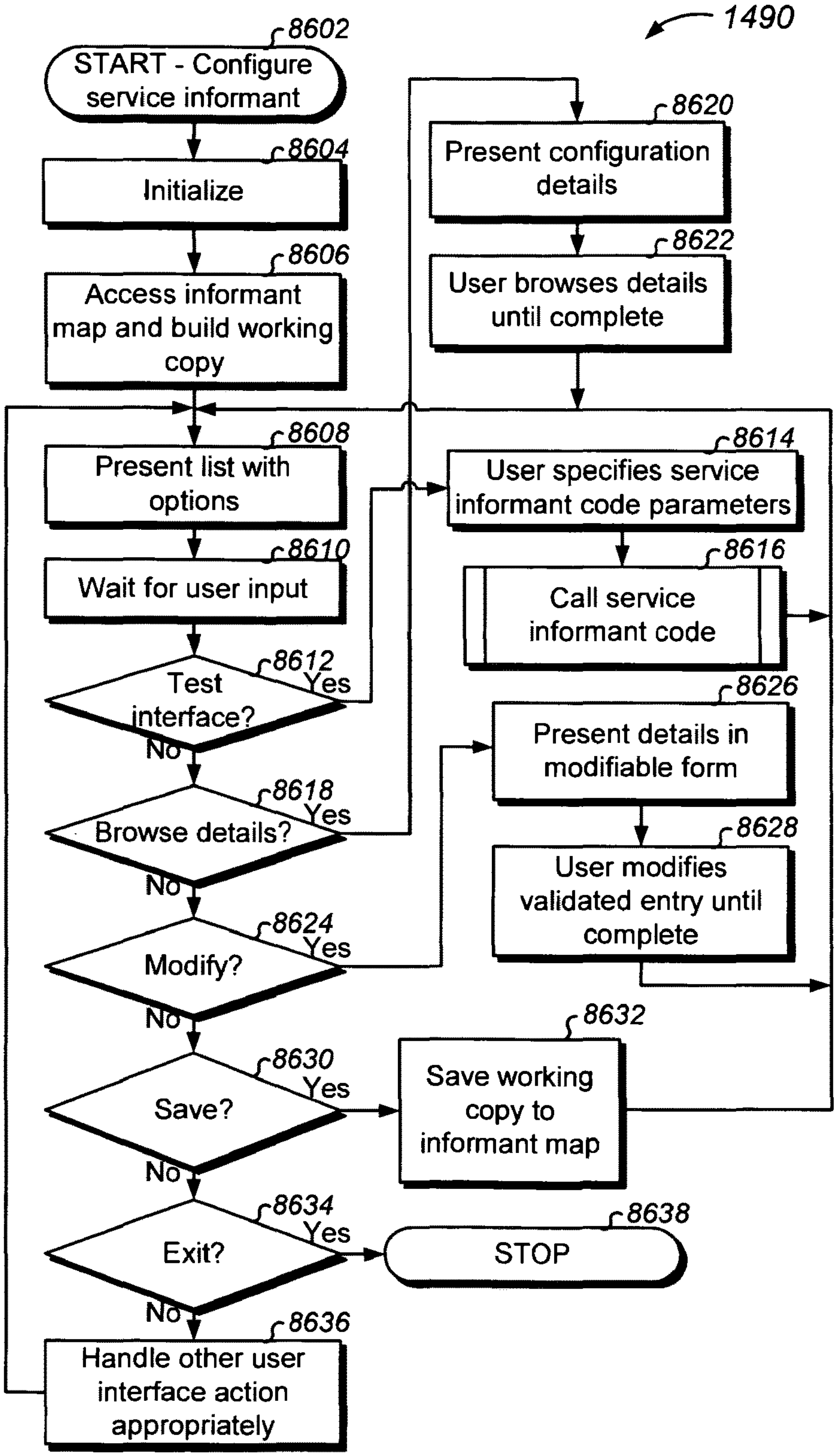

1490
8602
START - Configure service informant
8604
Initialize
8606
Access informant map and build working copy
8608
Present list with options
8610
Wait for user input
8612
Test interface?
Yes
No
8618
Browse details?
Yes
No
8624
Modify?
Yes
No
8630
Save?
Yes
No
8634
Exit?
Yes
No
8636
Handle other user interface action appropriately
8620
Present configuration details
8622
User browses details until complete
8614
User specifies service informant code parameters
8616
Call service informant code
8626
Present details in modifiable form
8628
User modifies validated entry until complete
8632
Save working copy to informant map
8638
STOP

Fig. 86A

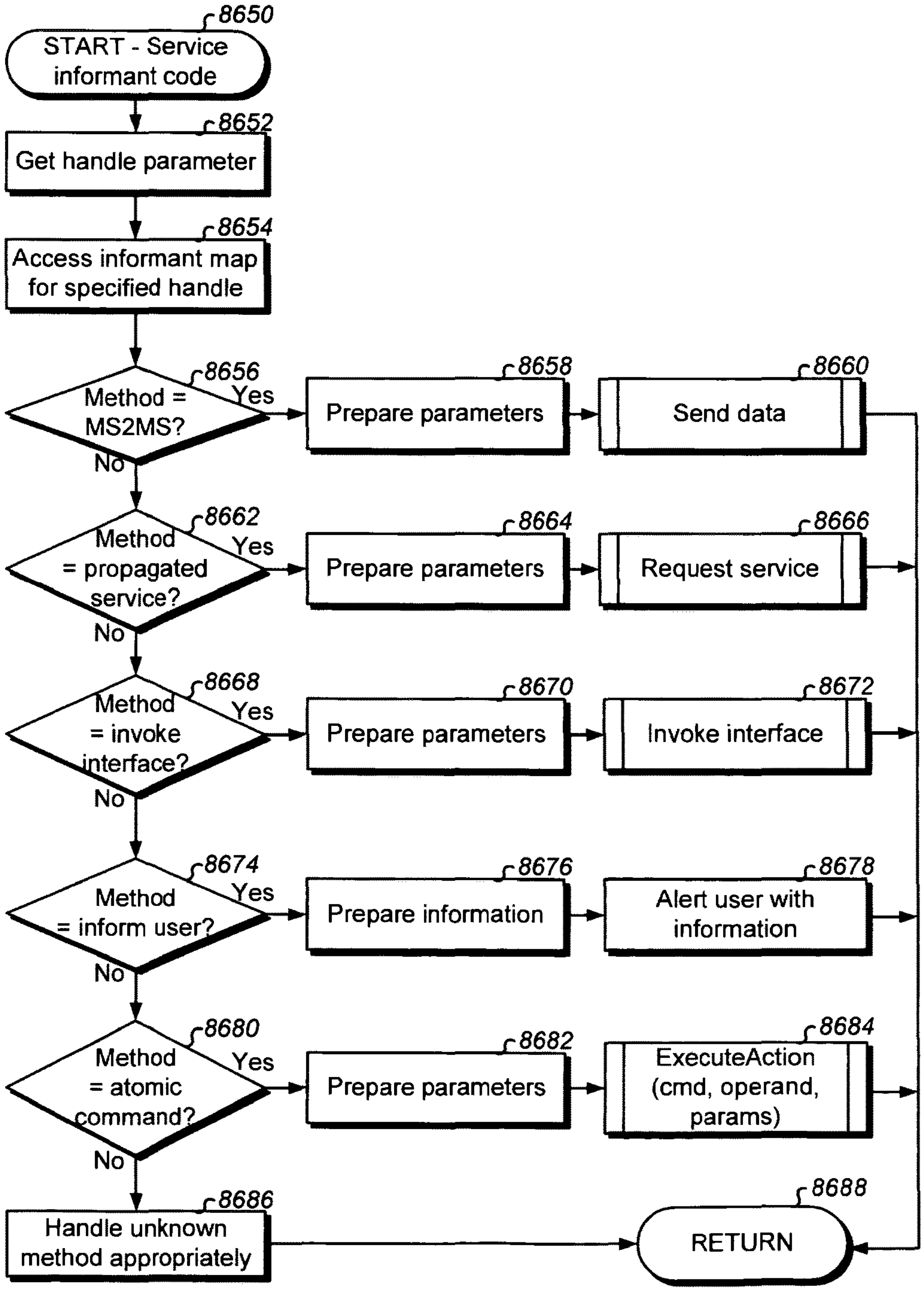

8650
START - Service informant code
8652
Get handle parameter
8654
Access informant map for specified handle
8656
Method = MS2MS?
Yes
No
8658
Prepare parameters
8660
Send data
8662
Method = propagated service?
Yes
No
8664
Prepare parameters
8666
Request service
8668
Method = invoke interface?
Yes
No
8670
Prepare parameters
8672
Invoke interface
8674
Method = inform user?
Yes
No
8676
Prepare information
8678
Alert user with information
8680
Method = atomic command?
Yes
No
8682
Prepare parameters
8684
ExecuteAction (cmd, operand, params)
8686
Handle unknown method appropriately
8688
RETURN

Fig. 86B

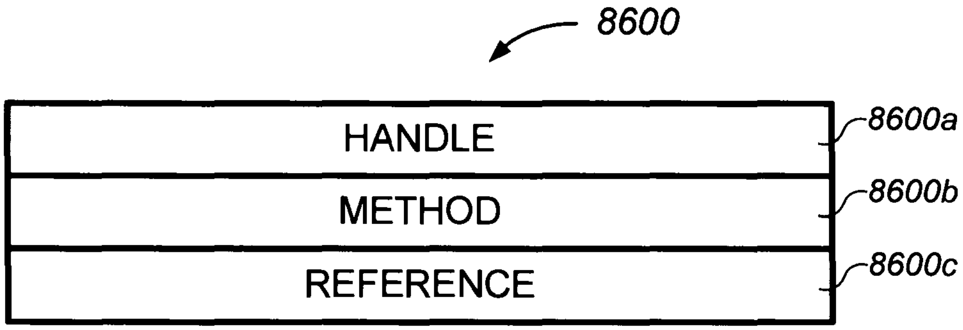


Fig. 86C

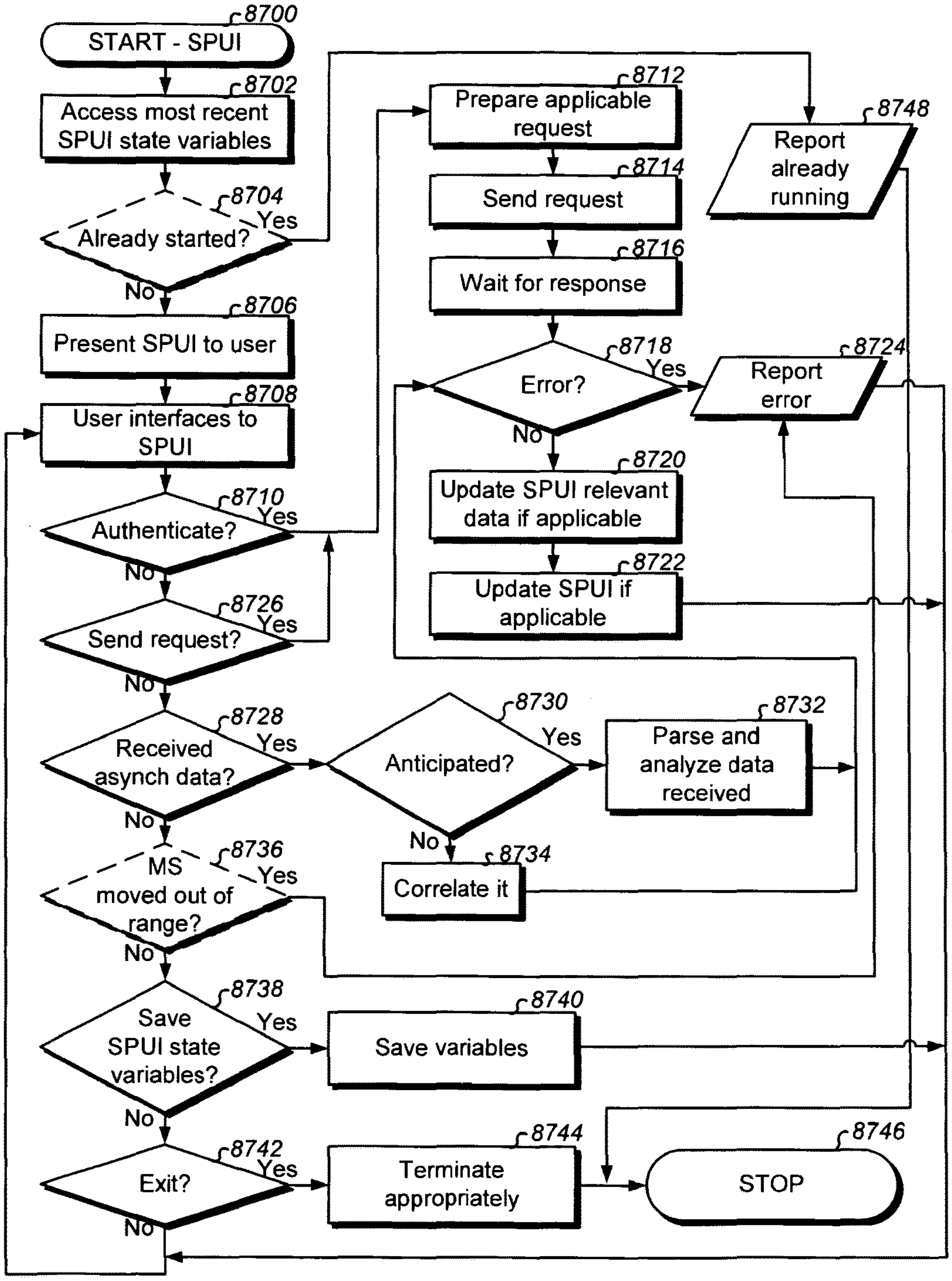

8700
START - SPUI
8702
Access most recent SPUI state variables
8704
Already started?
Yes
No
8706
Present SPUI to user
8708
User interfaces to SPUI
8710
Authenticate?
Yes
No
8726
Send request?
Yes
No
8728
Received asynch data?
Yes
No
8736
MS moved out of range?
Yes
No
8738
Save SPUI state variables?
Yes
No
8742
Exit?
Yes
No
8712
Prepare applicable request
8714
Send request
8716
Wait for response
8718
Error?
Yes
No
8720
Update SPUI relevant data if applicable
8722
Update SPUI if applicable
8748
Report already running
8724
Report error
8730
Anticipated?
Yes
No
8732
Parse and analyze data received
8734
Correlate it
8740
Save variables
8744
Terminate appropriately
8746
STOP

Fig. 87A

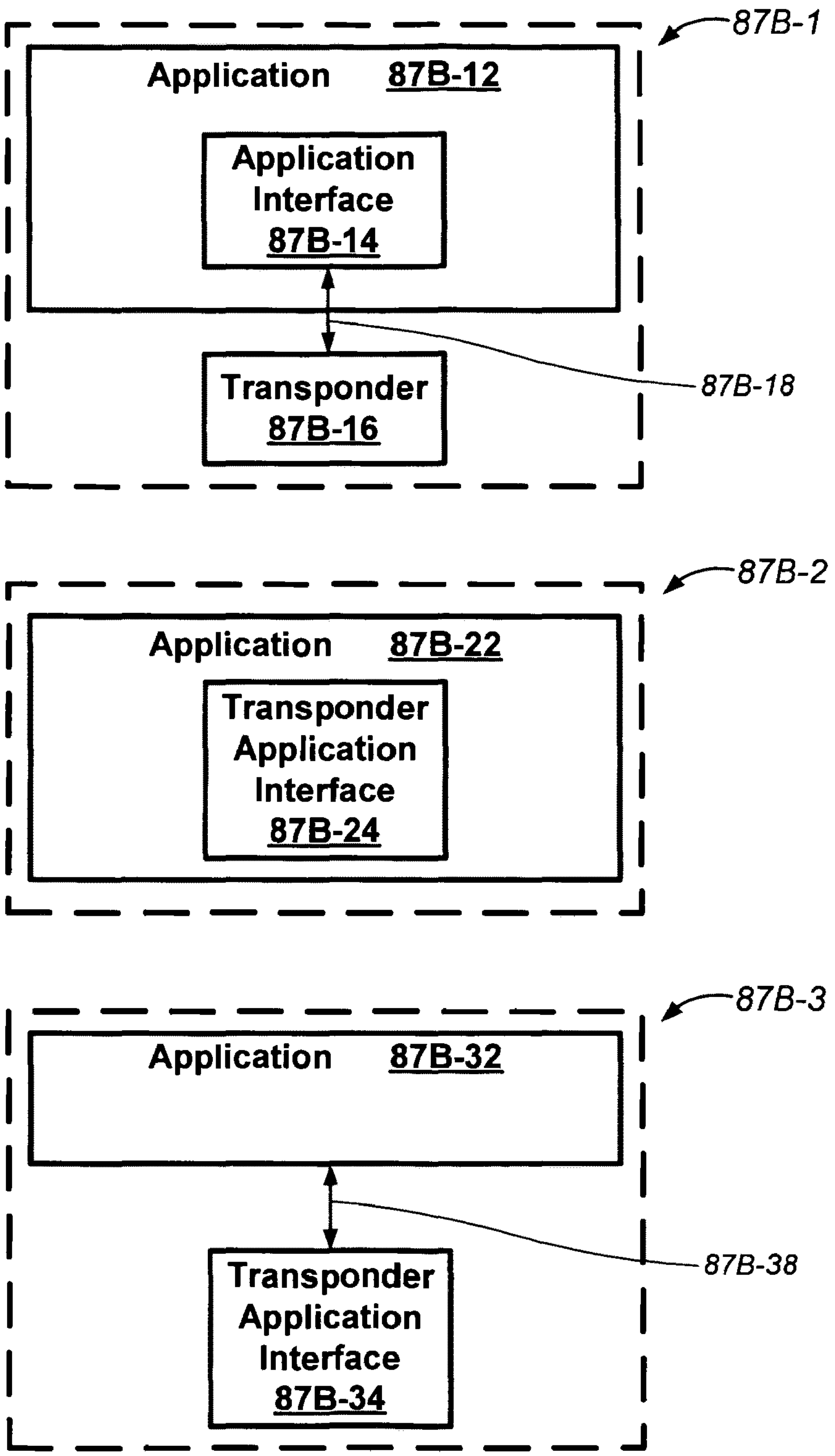

87B-1
Application 87B-12
Application Interface 87B-14
Transponder 87B-16
87B-18
87B-2
Application 87B-22
Transponder Application Interface 87B-24
87B-3
Application 87B-32
87B-38
Transponder Application Interface 87B-34

Fig. 87B

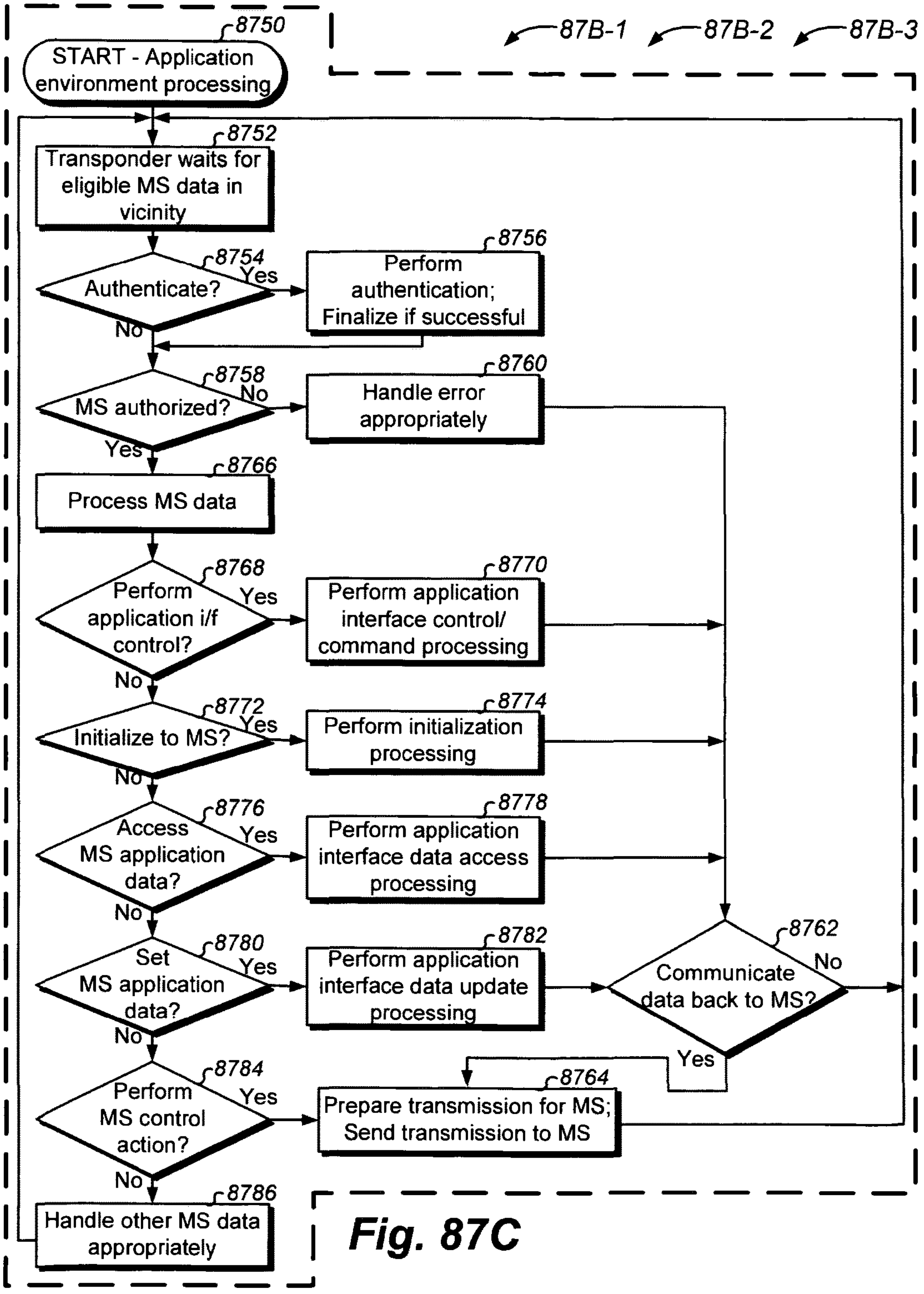

87B-1
87B-2
87B-3
8750
START - Application environment processing
8752
Transponder waits for eligible MS data in vicinity
8754
Authenticate?
Yes
No
8756
Perform authentication; Finalize if successful
8758
MS authorized?
No
Yes
8760
Handle error appropriately
8766
Process MS data
8768
Perform application i/f control?
Yes
No
8770
Perform application interface control/ command processing
8772
Initialize to MS?
Yes
No
8774
Perform initialization processing
8776
Access MS application data?
Yes
No
8778
Perform application interface data access processing
8780
Set MS application data?
Yes
No
8782
Perform application interface data update processing
8762
Communicate data back to MS?
No
Yes
8784
Perform MS control action?
Yes
No
8764
Prepare transmission for MS; Send transmission to MS
8786
Handle other MS data appropriately

Fig. 87C
]

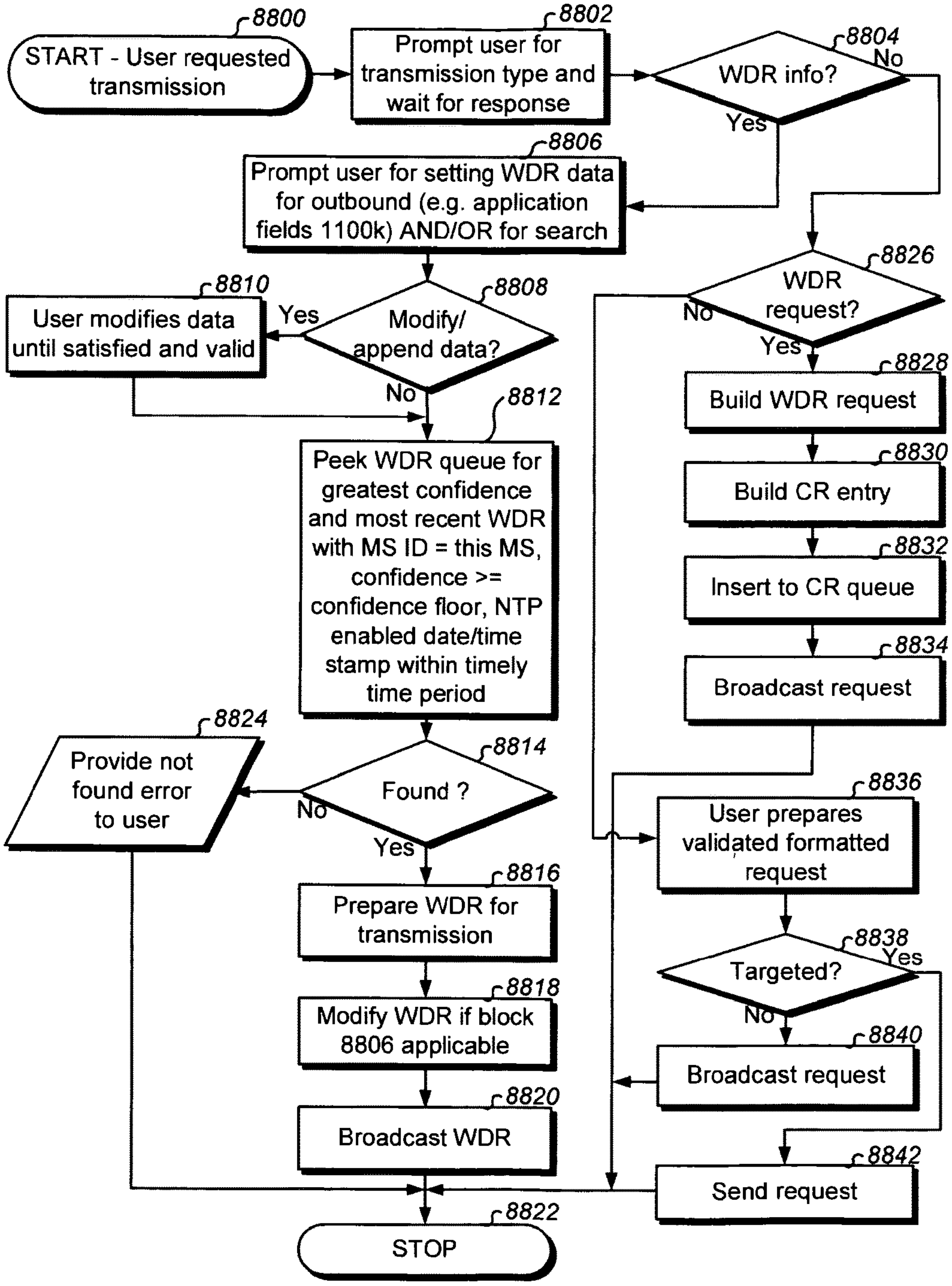

8800
START - User requested transmission
8802
Prompt user for transmission type and wait for response
8804
WDR info?
No
Yes
8806
Prompt user for setting WDR data for outbound (e.g. application fields 1100k) AND/OR for search
8810
User modifies data until satisfied and valid
Yes
8808
Modify/ append data?
No
8812
Peek WDR queue for greatest confidence and most recent WDR with MS ID = this MS, confidence >= confidence floor, NTP enabled date/time stamp within timely time period
8824
Provide not found error to user
8814
Found ?
No
Yes
8816
Prepare WDR for transmission
8818
Modify WDR if block 8806 applicable
8820
Broadcast WDR
8822
STOP
8826
WDR request?
No
Yes
8828
Build WDR request
8830
Build CR entry
8832
Insert to CR queue
8834
Broadcast request
8836
User prepares validated formatted request
8838
Targeted?
Yes
No
8840
Broadcast request
8842
Send request

Fig. 88A

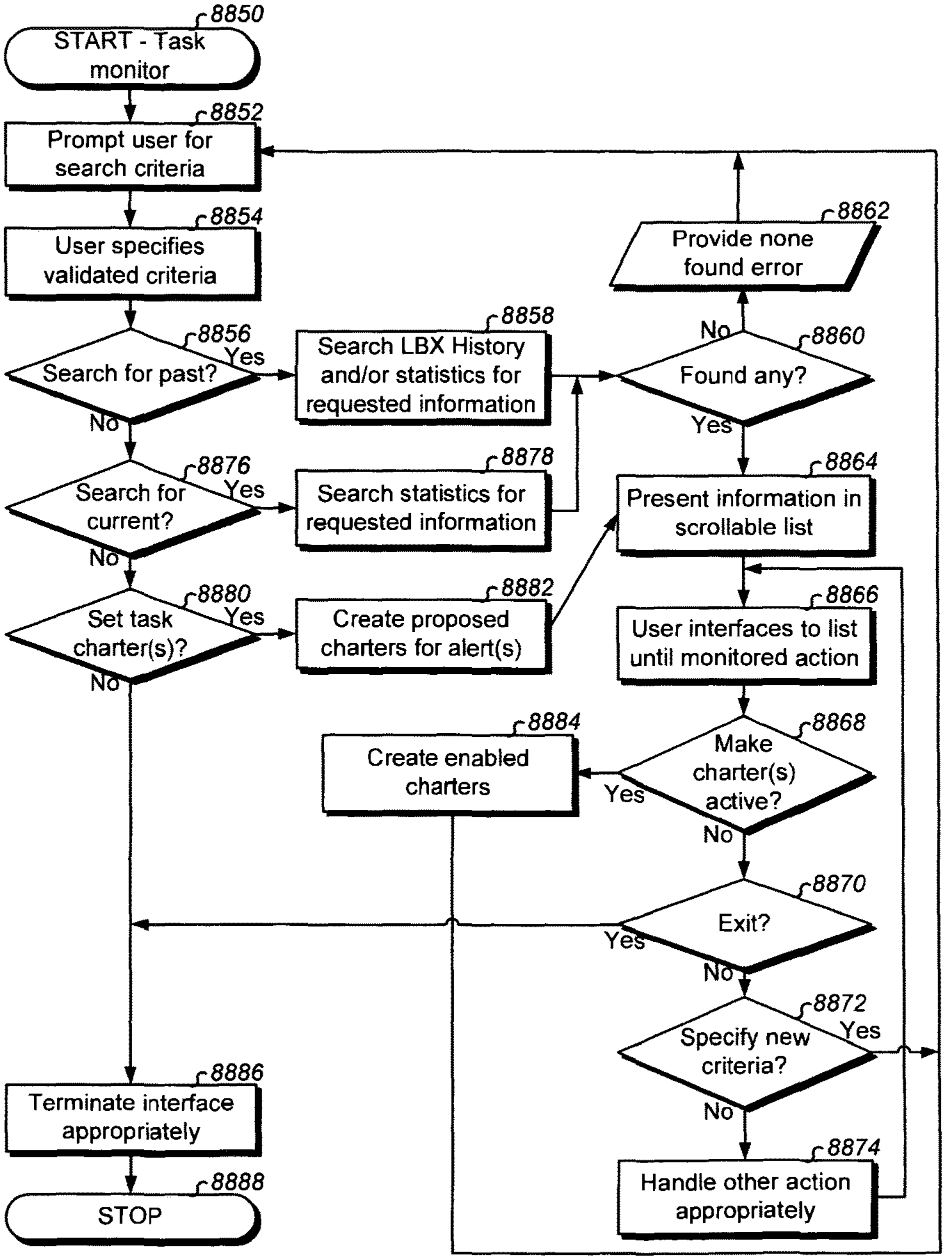

8850
START - Task monitor
8852
Prompt user for search criteria
8854
User specifies validated criteria
8856
Search for past?
Yes
No
8858
Search LBX History and/or statistics for requested information
8860
Found any?
No
Yes
8862
Provide none found error
8864
Present information in scrollable list
8876
Search for current?
Yes
No
8878
Search statistics for requested information
8880
Set task charter(s)?
Yes
No
8882
Create proposed charters for alert(s)
8866
User interfaces to list until monitored action
8868
Make charter(s) active?
Yes
No
8884
Create enabled charters
8870
Exit?
Yes
No
8872
Specify new criteria?
Yes
No
8874
Handle other action appropriately
8886
Terminate interface appropriately
8888
STOP

Fig. 88B

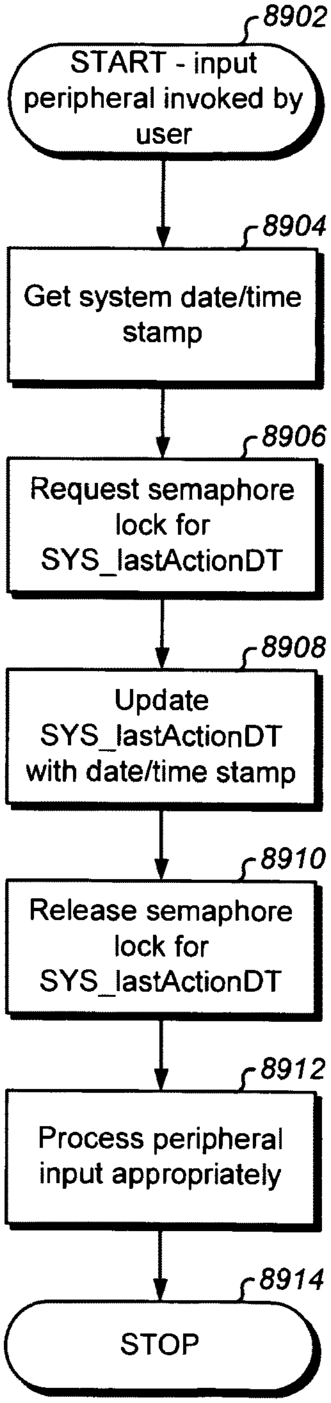

8902
START - input peripheral invoked by user
8904
Get system date/time stamp
8906
Request semaphore lock for SYS_lastActionDT
8908
Update SYS_lastActionDT with date/time stamp
8910
Release semaphore lock for SYS_lastActionDT
8912
Process peripheral input appropriately
8914
STOP

Fig. 89A

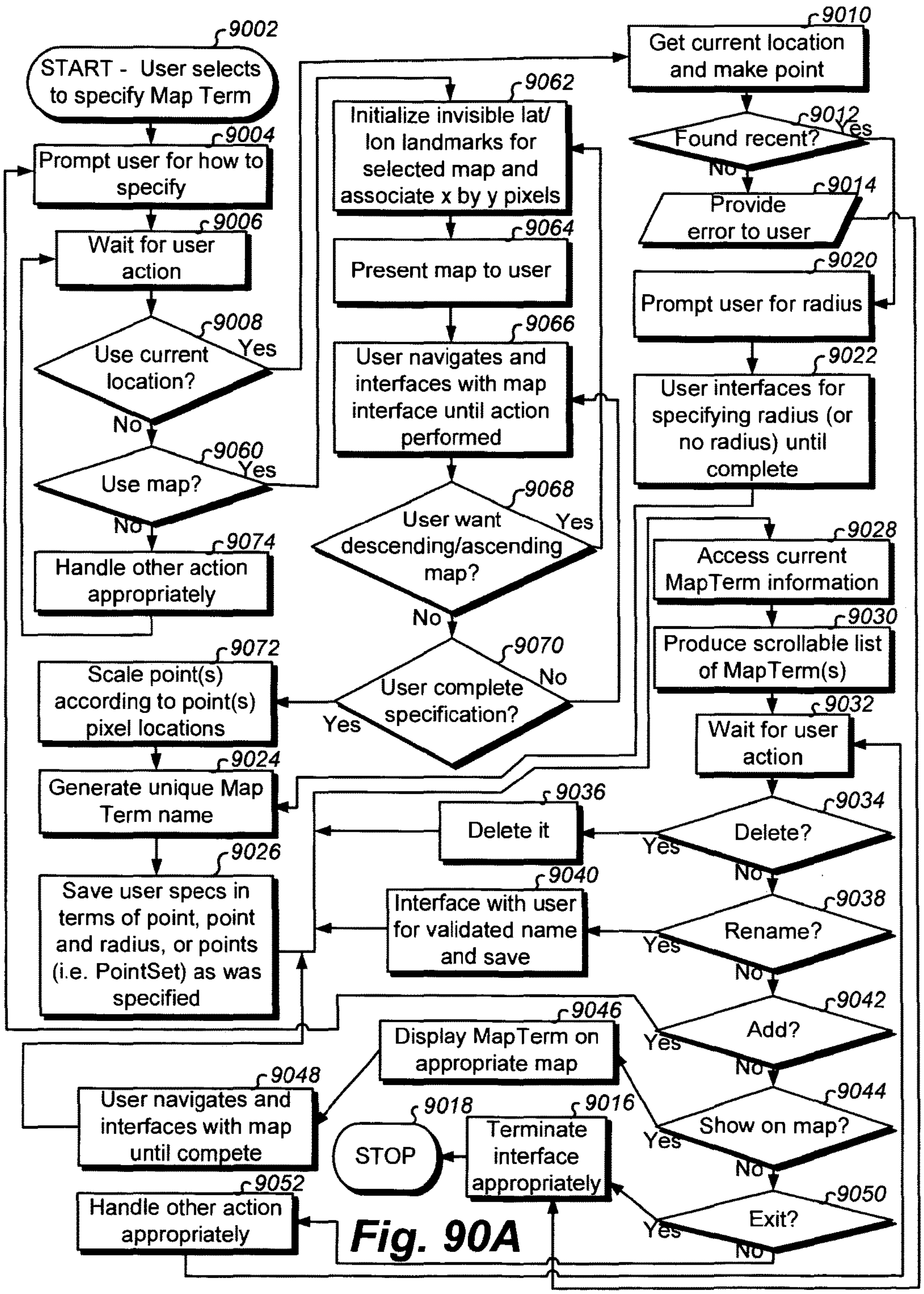

9002
START - User selects to specify Map Term
9004
Prompt user for how to specify
9006
Wait for user action
9008
Use current location?
Yes
No
9060
Use map?
Yes
No
9074
Handle other action appropriately
9062
Initialize invisible lat/ lon landmarks for selected map and associate x by y pixels
9064
Present map to user
9066
User navigates and interfaces with map interface until action performed
9068
User want descending/ascending map?
Yes
No
9070
User complete specification?
No
Yes
9072
Scale point(s) according to point(s) pixel locations
9024
Generate unique Map Term name
9026
Save user specs in terms of point, point and radius, or points (i.e. PointSet) as was specified
9010
Get current location and make point
9012
Found recent?
Yes
No
9014
Provide error to user
9020
Prompt user for radius
9022
User interfaces for specifying radius (or no radius) until complete
9028
Access current MapTerm information
9030
Produce scrollable list of MapTerm(s)
9032
Wait for user action
9034
Delete?
Yes
No
9036
Delete it
9038
Rename?
Yes
No
9040
Interface with user for validated name and save
9042
Add?
Yes
No
9046
Display MapTerm on appropriate map
9044
Show on map?
Yes
No
9048
User navigates and interfaces with map until compete
9050
Exit?
Yes
No
9016
Terminate interface appropriately
9018
STOP
9052
Handle other action appropriately

Fig. 90A

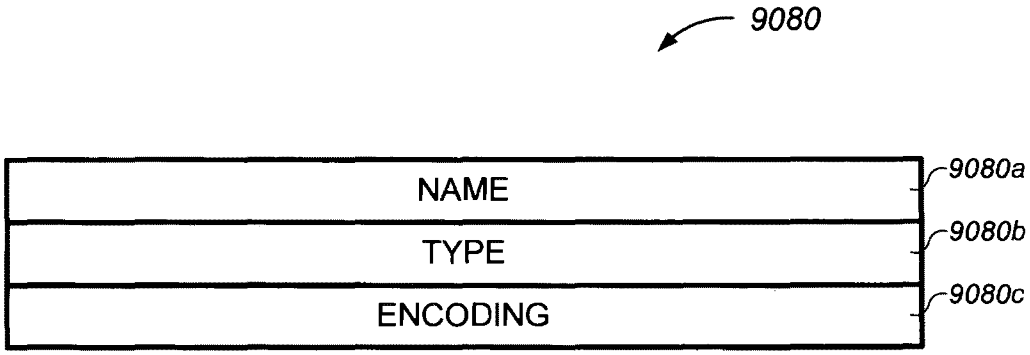


Fig. 90B

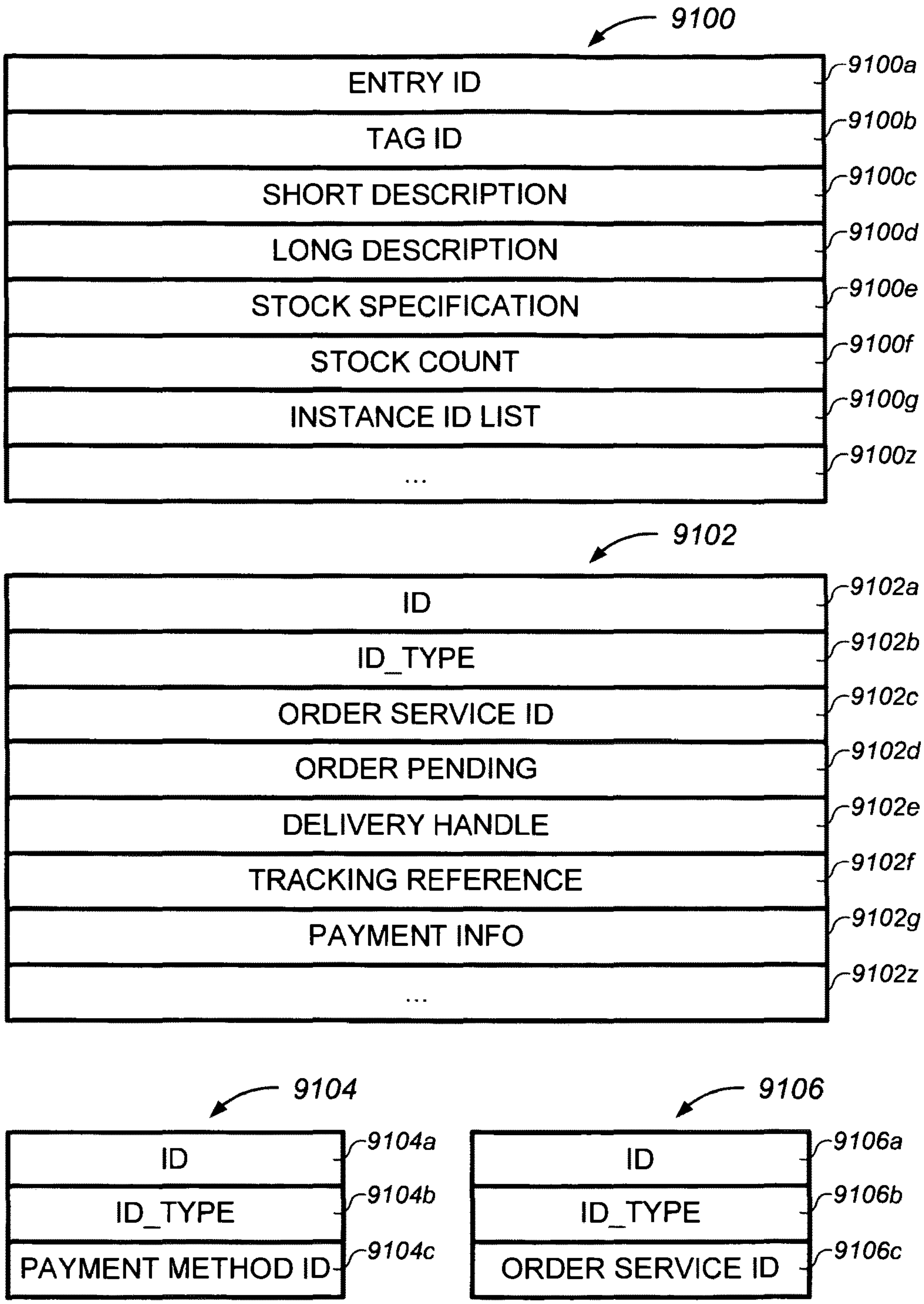

9100
ENTRY ID
9100a
TAG ID
9100b
SHORT DESCRIPTION
9100c
LONG DESCRIPTION
9100d
STOCK SPECIFICATION
9100e
STOCK COUNT
9100f
INSTANCE ID LIST
9100g
...
9100z
9102
ID
9102a
ID_TYPE
9102b
ORDER SERVICE ID
9102c
ORDER PENDING
9102d
DELIVERY HANDLE
9102e
TRACKING REFERENCE
9102f
PAYMENT INFO
9102g
...
9102z
9104
ID
9104a
ID_TYPE
9104b
PAYMENT METHOD ID
9104c
9106
ID
9106a
ID_TYPE
9106b
ORDER SERVICE ID
9106c

Fig. 91A

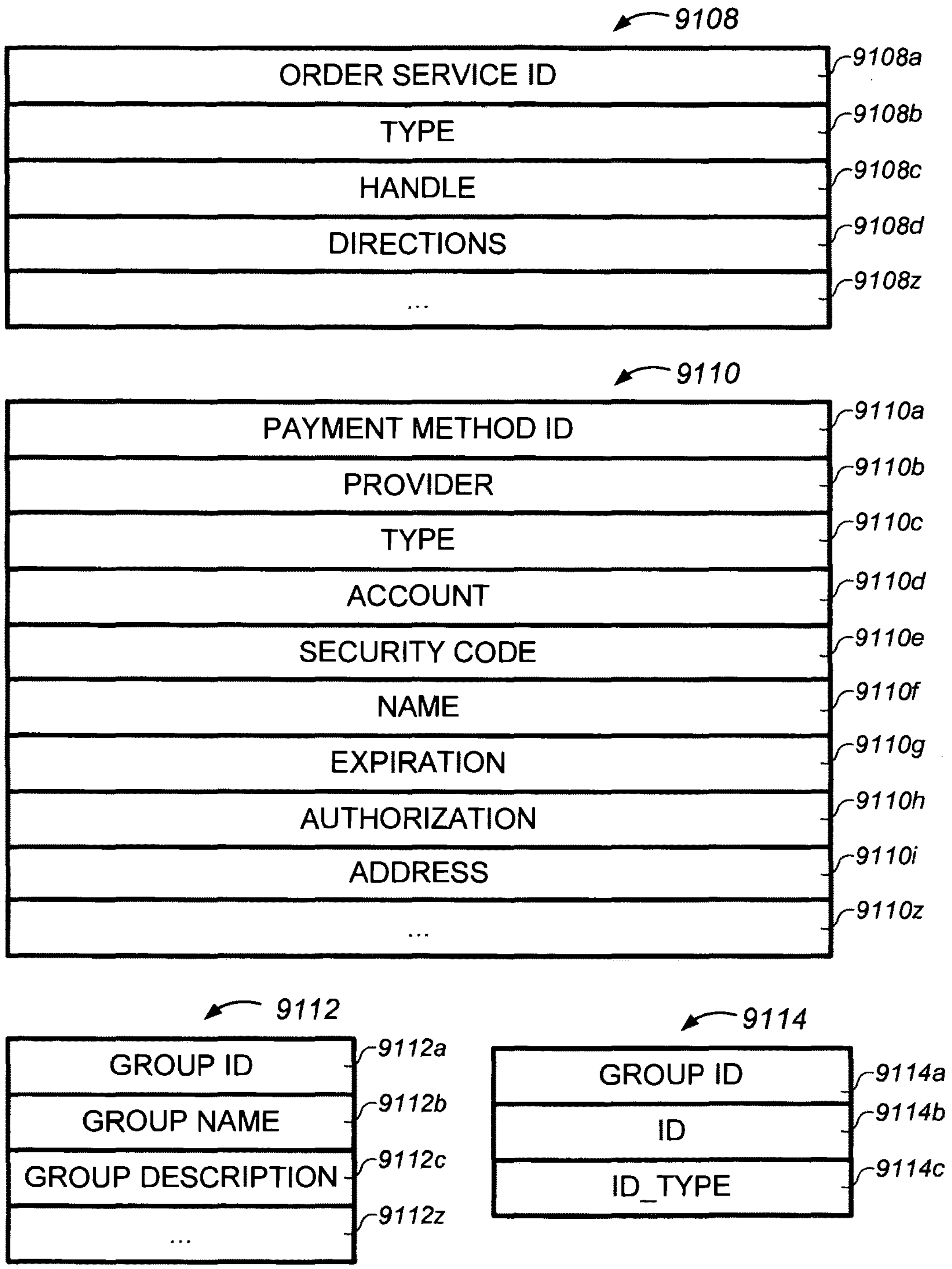

9108
ORDER SERVICE ID
9108a
TYPE
9108b
HANDLE
9108c
DIRECTIONS
9108d
...
9108z
9110
PAYMENT METHOD ID
9110a
PROVIDER
9110b
TYPE
9110c
ACCOUNT
9110d
SECURITY CODE
9110e
NAME
9110f
EXPIRATION
9110g
AUTHORIZATION
9110h
ADDRESS
9110i
...
9110z
9112
GROUP ID
9112a
GROUP NAME
9112b
GROUP DESCRIPTION
9112c
...
9112z
9114
GROUP ID
9114a
ID
9114b
ID_TYPE
9114c

Fig. 91B

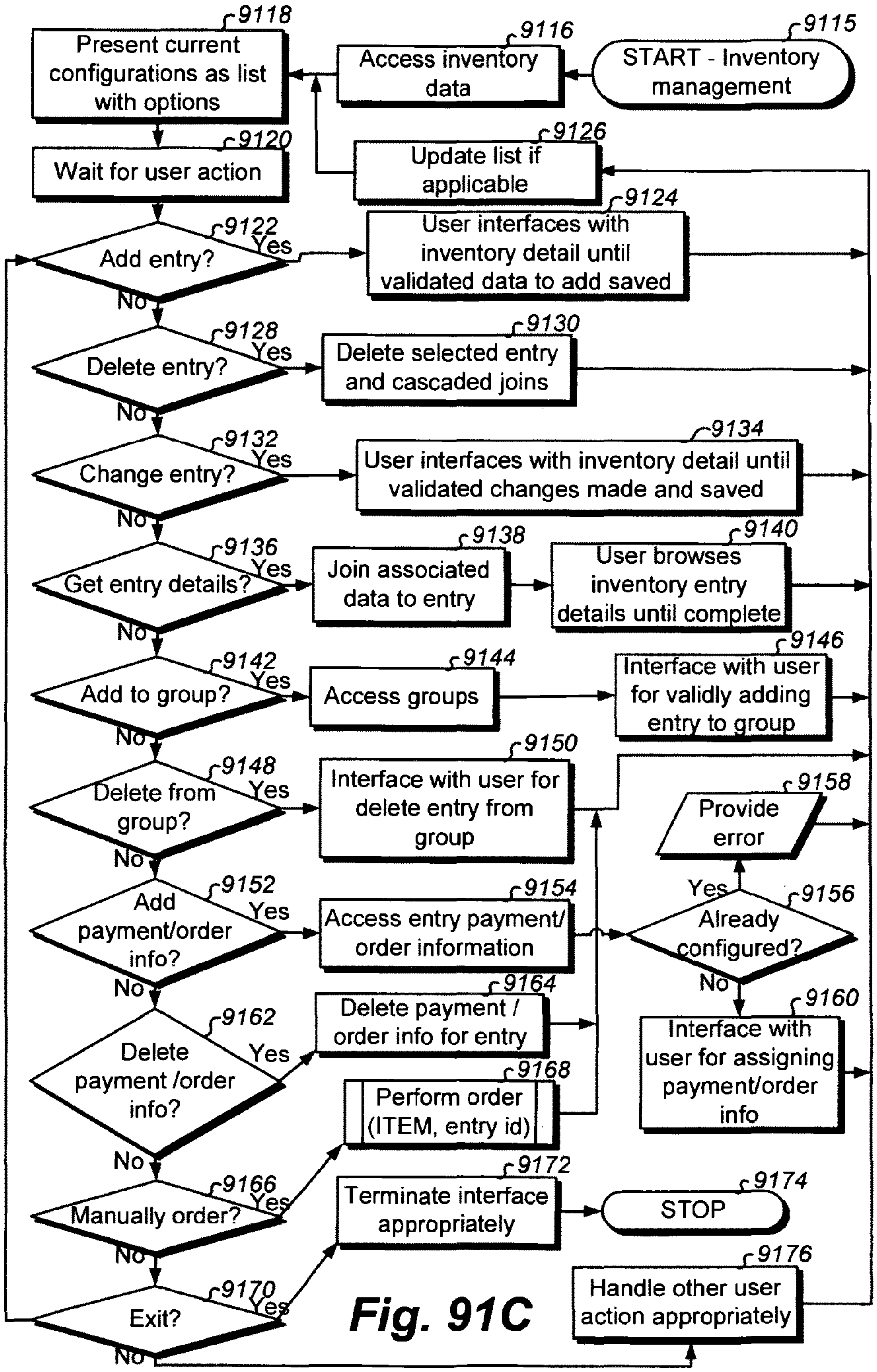

9115
START - Inventory management
9116
Access inventory data
9118
Present current configurations as list with options
9120
Wait for user action
9126
Update list if applicable
9122
Add entry?
Yes
No
9124
User interfaces with inventory detail until validated data to add saved
9128
Delete entry?
Yes
No
9130
Delete selected entry and cascaded joins
9132
Change entry?
Yes
No
9134
User interfaces with inventory detail until validated changes made and saved
9136
Get entry details?
Yes
No
9138
Join associated data to entry
9140
User browses inventory entry details until complete
9142
Add to group?
Yes
No
9144
Access groups
9146
Interface with user for validly adding entry to group
9148
Delete from group?
Yes
No
9150
Interface with user for delete entry from group
9158
Provide error
9152
Add payment/order info?
Yes
No
9154
Access entry payment/ order information
9156
Already configured?
Yes
No
9160
Interface with user for assigning payment/order info
9162
Delete payment /order info?
Yes
No
9164
Delete payment / order info for entry
9168
Perform order (ITEM, entry id)
9166
Manually order?
Yes
No
9172
Terminate interface appropriately
9174
STOP
9170
Exit?
Yes
No
9176
Handle other user action appropriately

Fig. 91C

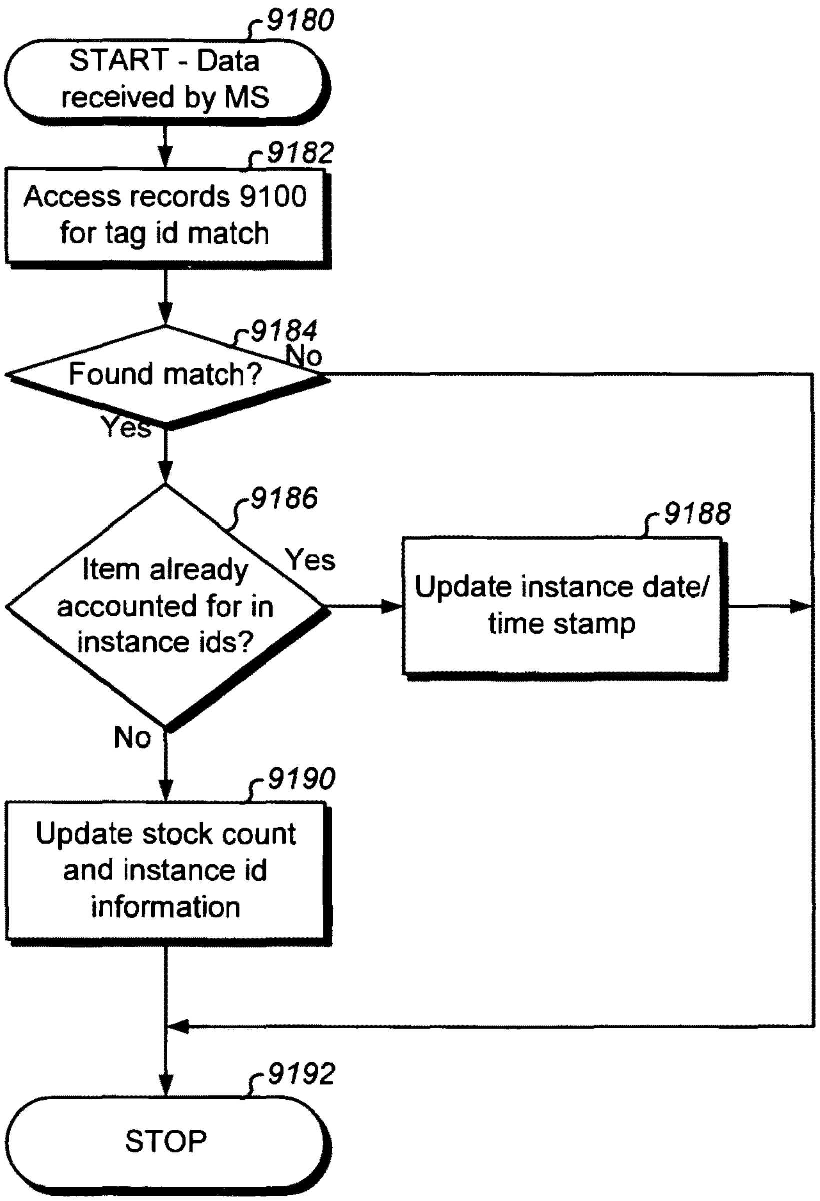

9180
START - Data received by MS
9182
Access records 9100 for tag id match
9184
Found match?
No
Yes
9186
Item already accounted for in instance ids?
Yes
9188
Update instance date/ time stamp
No
9190
Update stock count and instance id information
9192
STOP

Fig. 91D

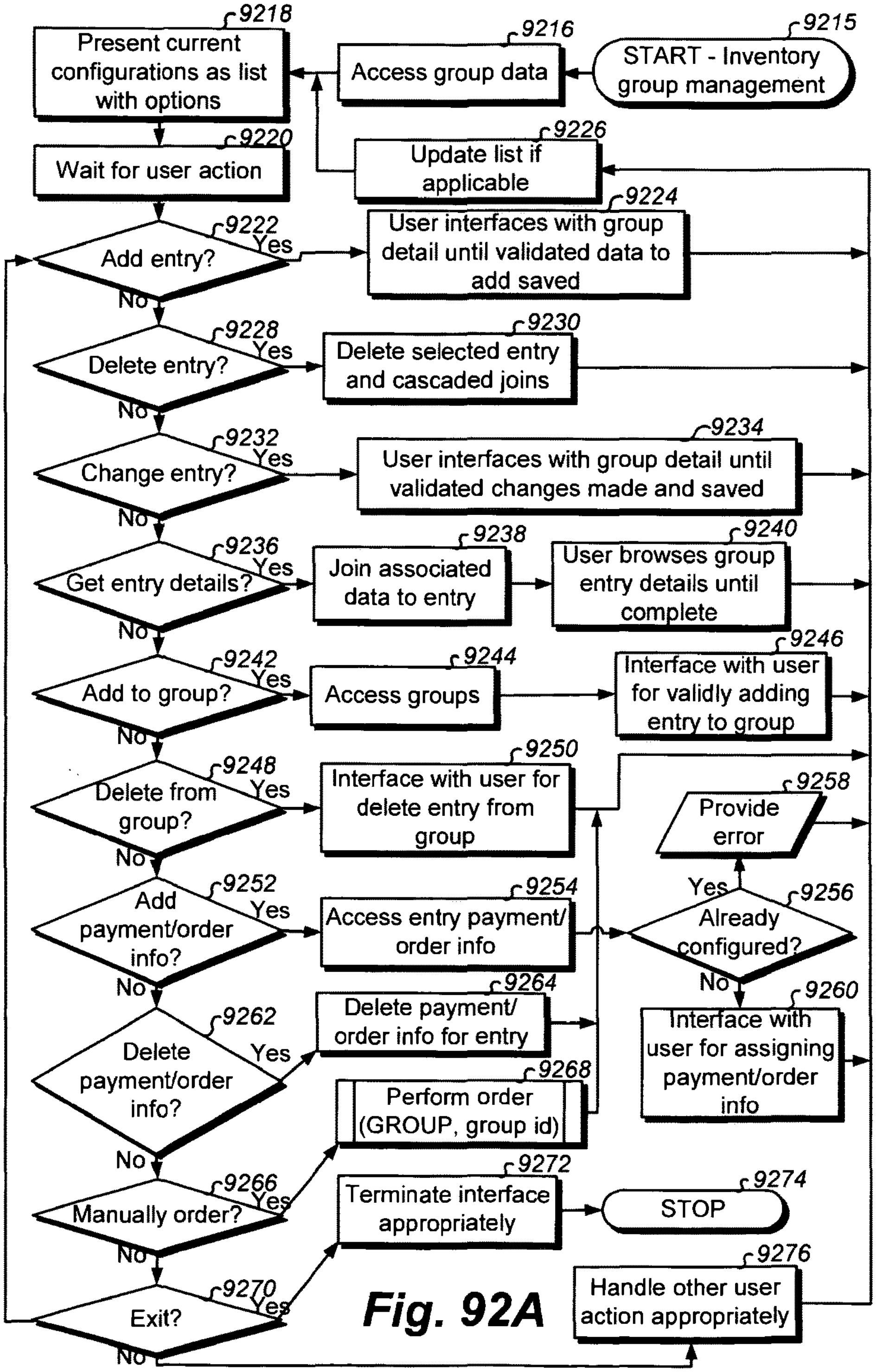

9215
START - Inventory group management
9216
Access group data
9218
Present current configurations as list with options
9220
Wait for user action
9226
Update list if applicable
9222
Add entry?
Yes
No
9224
User interfaces with group detail until validated data to add saved
9228
Delete entry?
Yes
No
9230
Delete selected entry and cascaded joins
9232
Change entry?
Yes
No
9234
User interfaces with group detail until validated changes made and saved
9236
Get entry details?
Yes
No
9238
Join associated data to entry
9240
User browses group entry details until complete
9242
Add to group?
Yes
No
9244
Access groups
9246
Interface with user for validly adding entry to group
9248
Delete from group?
Yes
No
9250
Interface with user for delete entry from group
9258
Provide error
9252
Add payment/order info?
Yes
No
9254
Access entry payment/ order info
9256
Already configured?
Yes
No
9260
Interface with user for assigning payment/order info
9262
Delete payment/order info?
Yes
No
9264
Delete payment/ order info for entry
9268
Perform order (GROUP, group id)
9266
Manually order?
Yes
No
9272
Terminate interface appropriately
9274
STOP
9270
Exit?
Yes
No
9276
Handle other user action appropriately

Fig. 92A

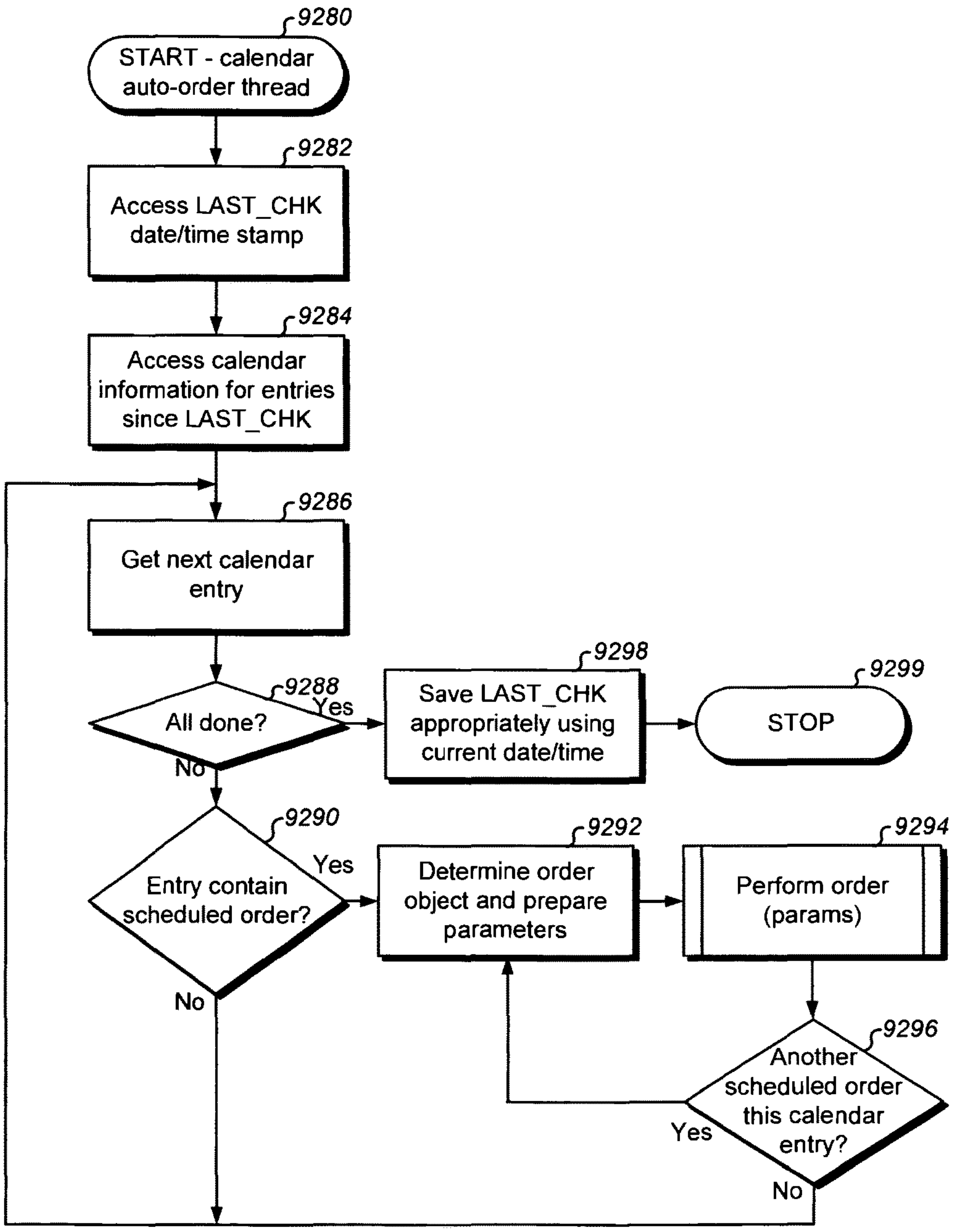

9280
START - calendar auto-order thread
9282
Access LAST_CHK date/time stamp
9284
Access calendar information for entries since LAST_CHK
9286
Get next calendar entry
9288
All done?
Yes
No
9298
Save LAST_CHK appropriately using current date/time
9299
STOP
9290
Entry contain scheduled order?
Yes
No
9292
Determine order object and prepare parameters
9294
Perform order (params)
9296
Another scheduled order this calendar entry?
Yes
No

Fig. 92B

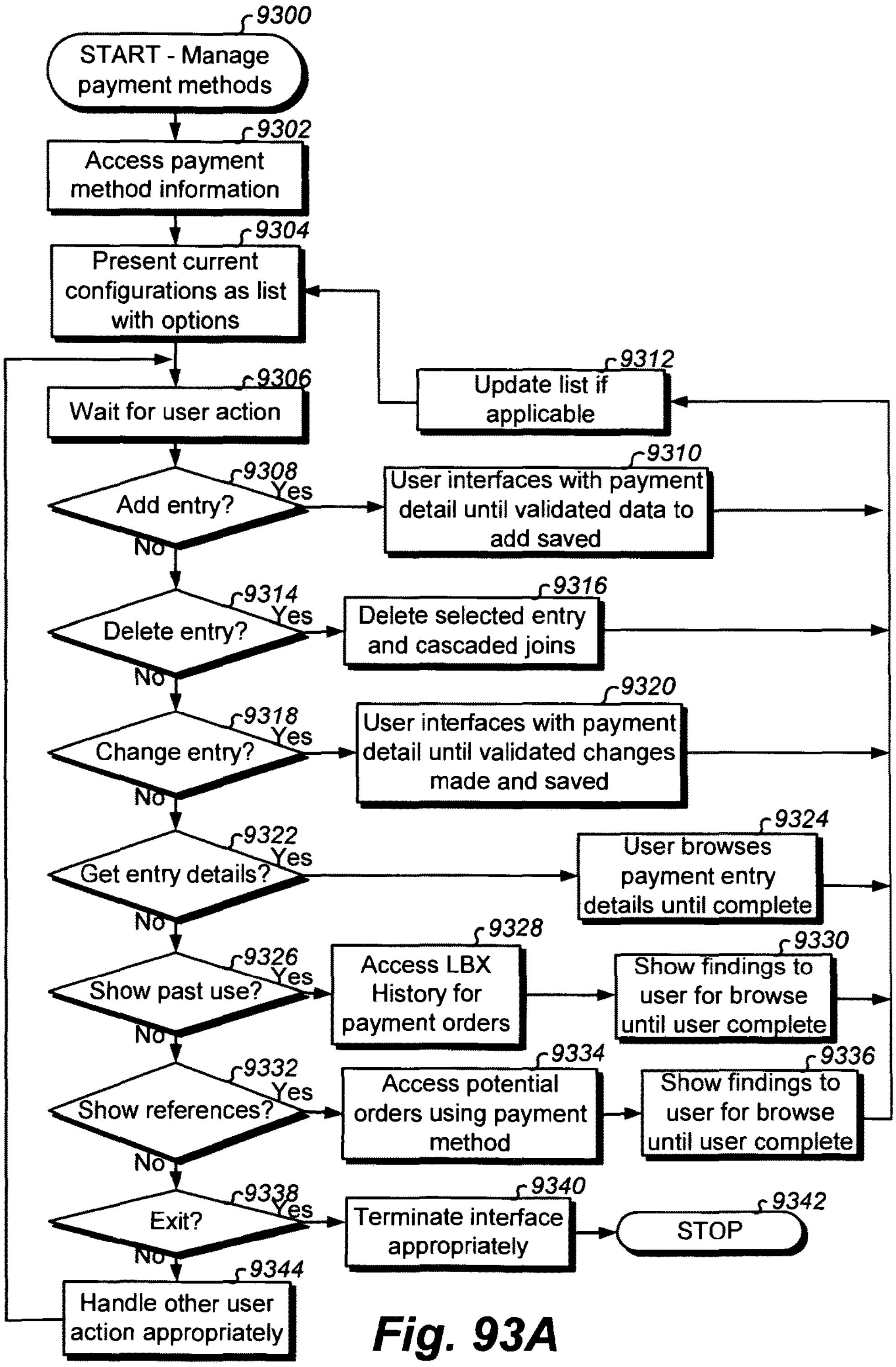

9300
START - Manage payment methods
9302
Access payment method information
9304
Present current configurations as list with options
9306
Wait for user action
9312
Update list if applicable
9308
Add entry?
Yes
No
9310
User interfaces with payment detail until validated data to add saved
9314
Delete entry?
Yes
No
9316
Delete selected entry and cascaded joins
9318
Change entry?
Yes
No
9320
User interfaces with payment detail until validated changes made and saved
9322
Get entry details?
Yes
No
9324
User browses payment entry details until complete
9326
Show past use?
Yes
No
9328
Access LBX History for payment orders
9330
Show findings to user for browse until user complete
9332
Show references?
Yes
No
9334
Access potential orders using payment method
9336
Show findings to user for browse until user complete
9338
Exit?
Yes
No
9340
Terminate interface appropriately
9342
STOP
9344
Handle other user action appropriately

Fig. 93A

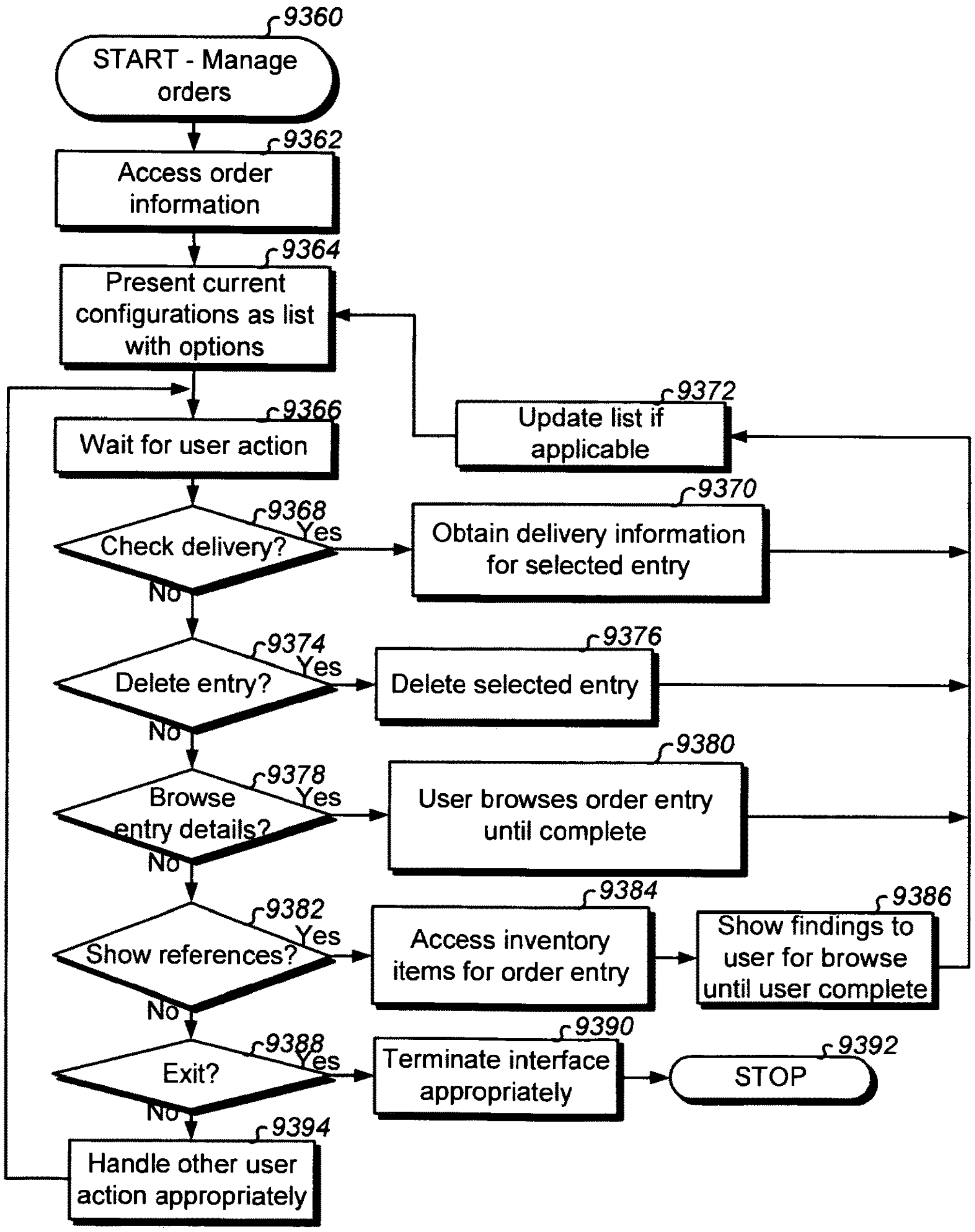

9360
START - Manage orders
9362
Access order information
9364
Present current configurations as list with options
9366
Wait for user action
9368
Check delivery?
Yes
No
9370
Obtain delivery information for selected entry
9372
Update list if applicable
9374
Delete entry?
Yes
No
9376
Delete selected entry
9378
Browse entry details?
Yes
No
9380
User browses order entry until complete
9382
Show references?
Yes
No
9384
Access inventory items for order entry
9386
Show findings to user for browse until user complete
9388
Exit?
Yes
No
9390
Terminate interface appropriately
9392
STOP
9394
Handle other user action appropriately

Fig. 93B

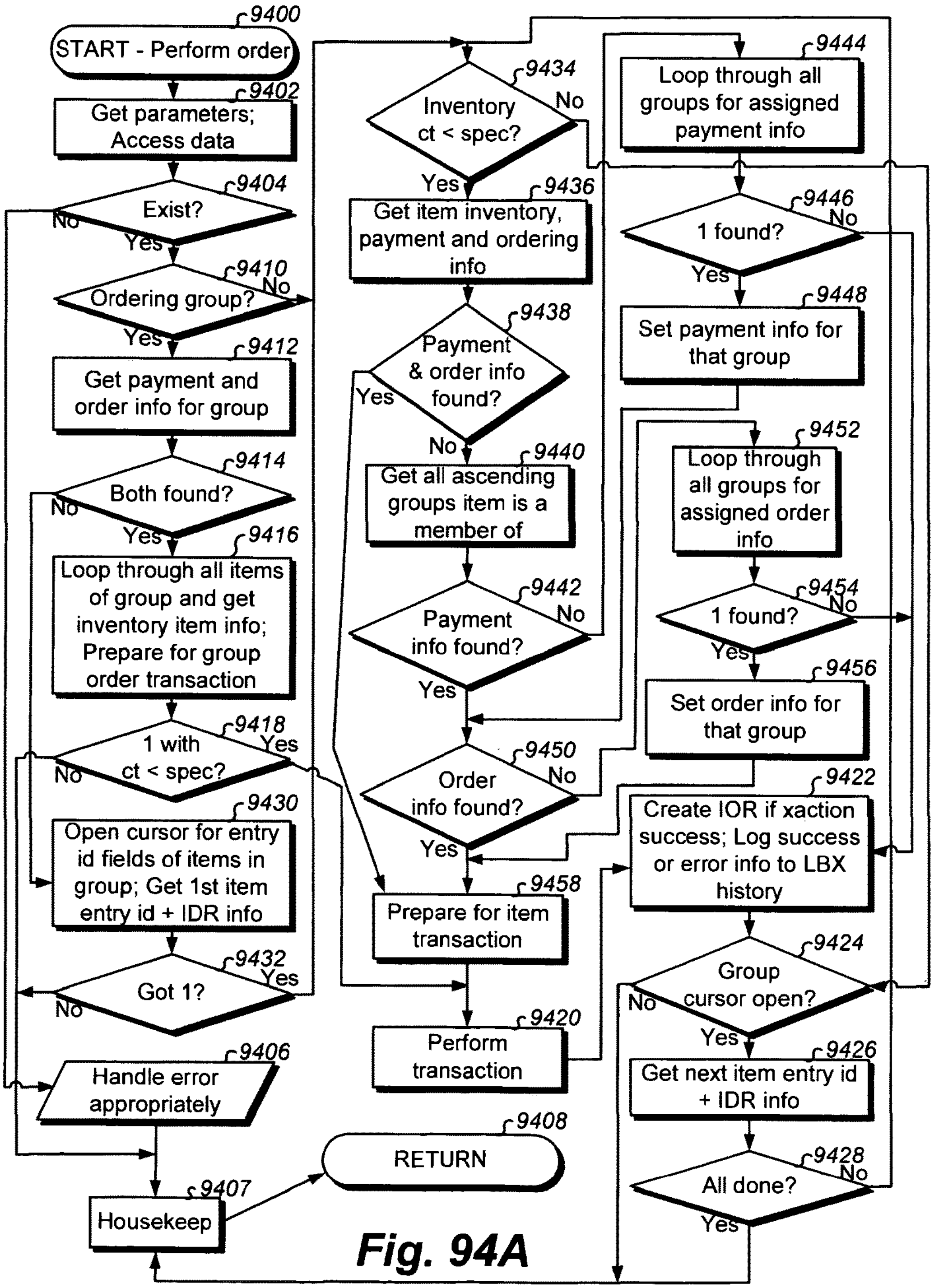

9400
START - Perform order
9402
Get parameters; Access data
9404
Exist?
No
Yes
9410
Ordering group?
No
Yes
9412
Get payment and order info for group
9414
Both found?
No
Yes
9416
Loop through all items of group and get inventory item info; Prepare for group order transaction
9418
1 with ct < spec?
Yes
No
9430
Open cursor for entry id fields of items in group; Get 1st item entry id + IDR info
9432
Got 1?
Yes
No
9406
Handle error appropriately
9407
Housekeep
9434
Inventory ct < spec?
No
Yes
9436
Get item inventory, payment and ordering info
9438
Payment & order info found?
Yes
No
9440
Get all ascending groups item is a member of
9442
Payment info found?
No
Yes
9450
Order info found?
No
Yes
9458
Prepare for item transaction
9420
Perform transaction
9408
RETURN
9444
Loop through all groups for assigned payment info
9446
1 found?
No
Yes
9448
Set payment info for that group
9452
Loop through all groups for assigned order info
9454
1 found?
No
Yes
9456
Set order info for that group
9422
Create IOR if xaction success; Log success or error info to LBX history
9424
Group cursor open?
No
Yes
9426
Get next item entry id + IDR info
9428
All done?
No
Yes

Fig. 94A

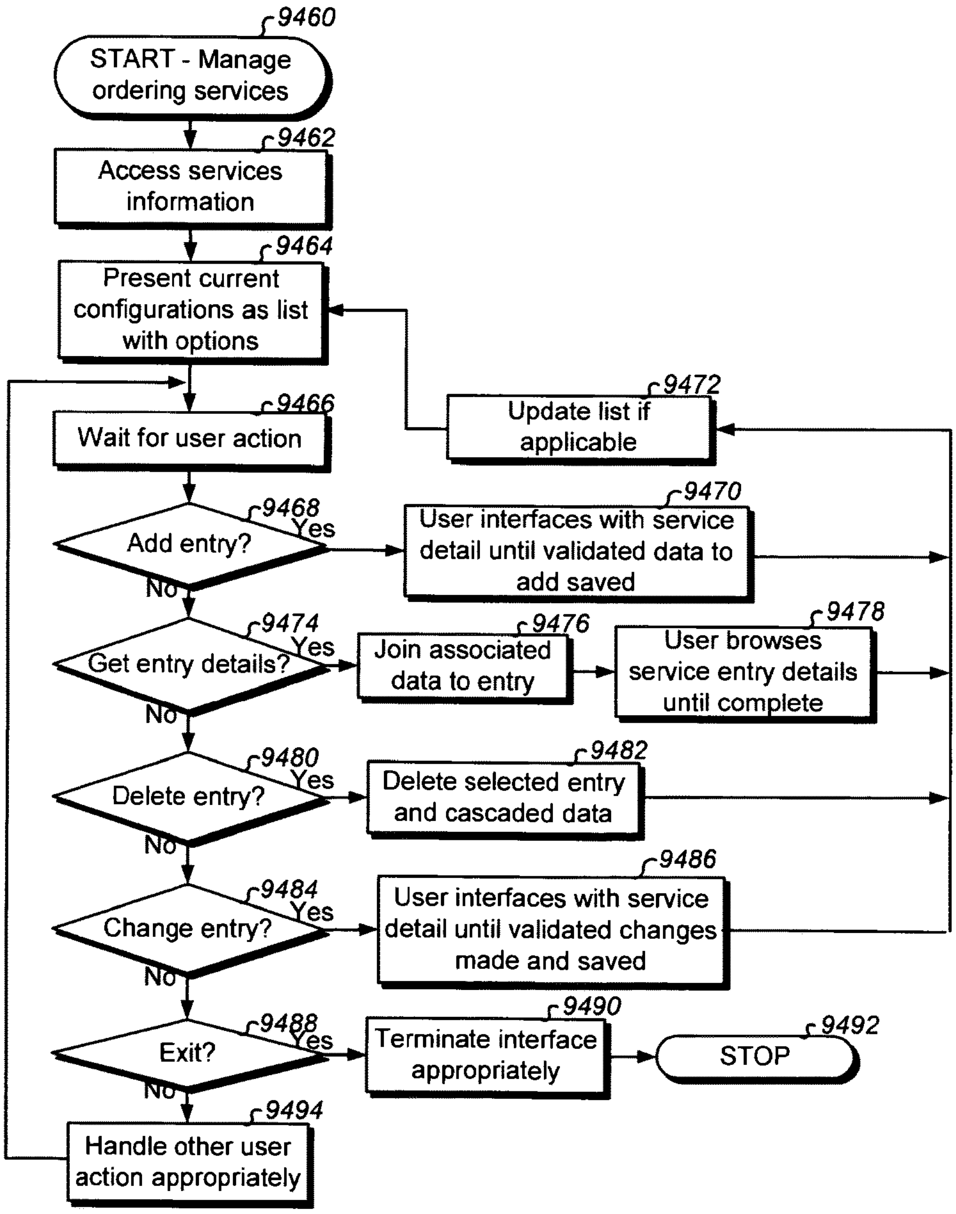

9460
START - Manage ordering services
9462
Access services information
9464
Present current configurations as list with options
9466
Wait for user action
9468
Add entry?
Yes
No
9470
User interfaces with service detail until validated data to add saved
9472
Update list if applicable
9474
Get entry details?
Yes
No
9476
Join associated data to entry
9478
User browses service entry details until complete
9480
Delete entry?
Yes
No
9482
Delete selected entry and cascaded data
9484
Change entry?
Yes
No
9486
User interfaces with service detail until validated changes made and saved
9488
Exit?
Yes
No
9490
Terminate interface appropriately
9492
STOP
9494
Handle other user action appropriately

Fig. 94B

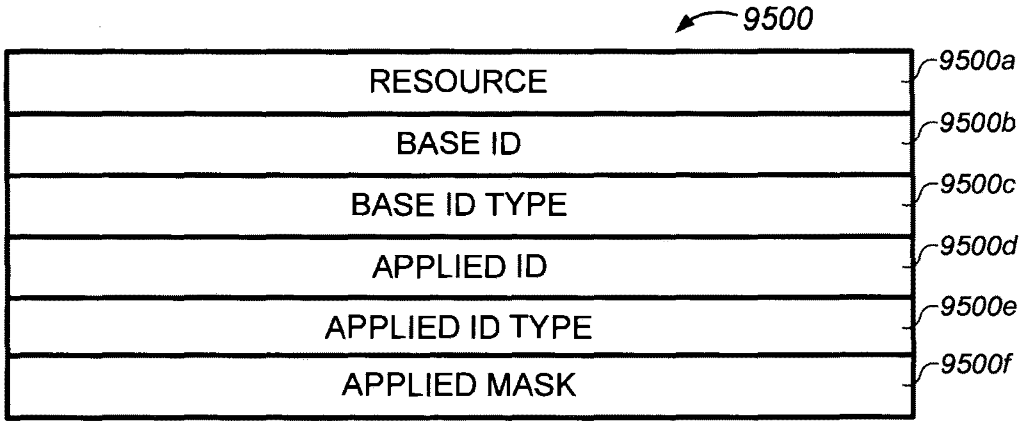

9500
RESOURCE
9500a
BASE ID
9500b
BASE ID TYPE
9500c
APPLIED ID
9500d
APPLIED ID TYPE
9500e
APPLIED MASK
9500f

Fig. 95A

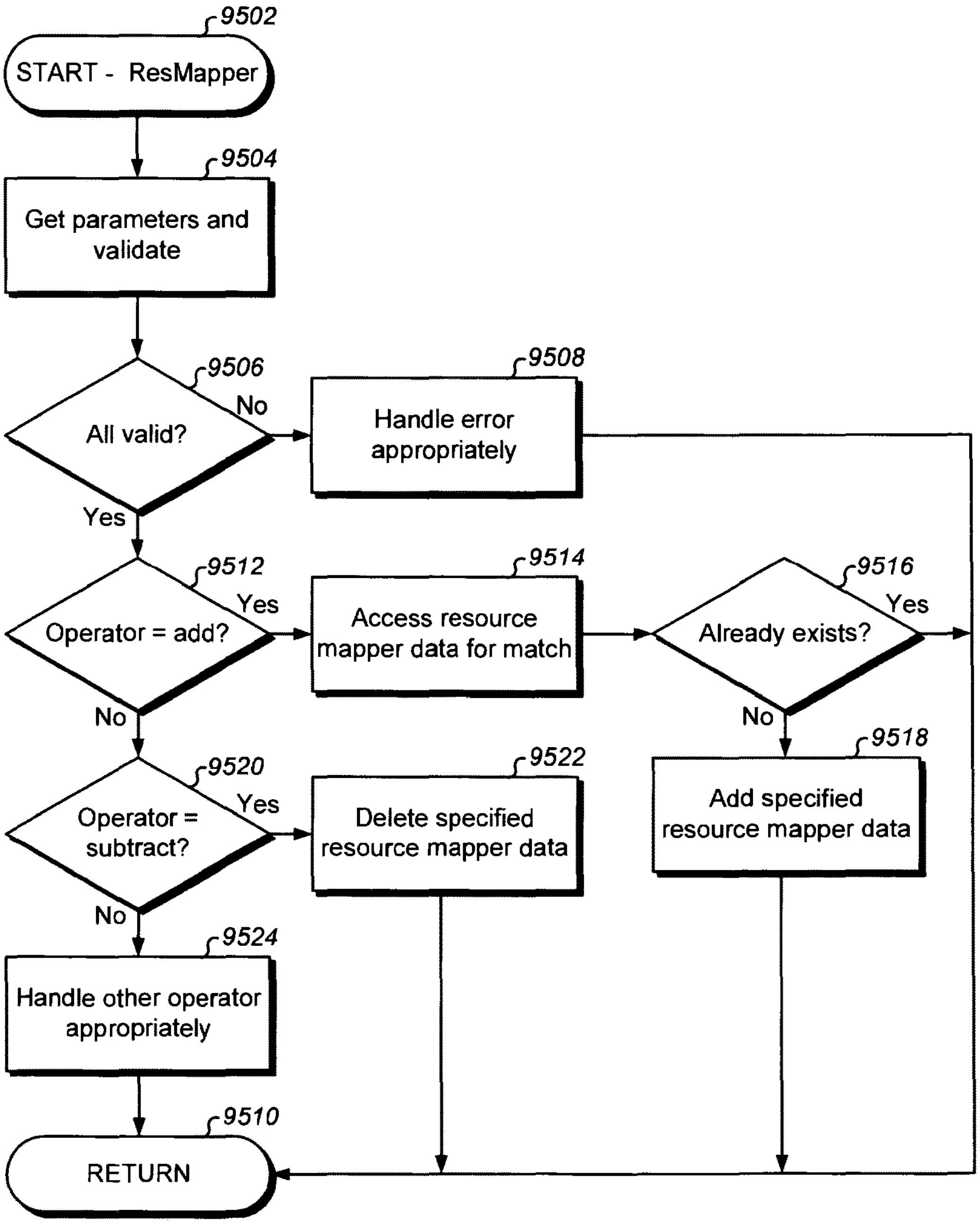

9502
START - ResMapper
9504
Get parameters and validate
9506
All valid?
No
Yes
9508
Handle error appropriately
9512
Operator = add?
Yes
No
9514
Access resource mapper data for match
9516
Already exists?
Yes
No
9518
Add specified resource mapper data
9520
Operator = subtract?
Yes
No
9522
Delete specified resource mapper data
9524
Handle other operator appropriately
9510
RETURN

Fig. 95B

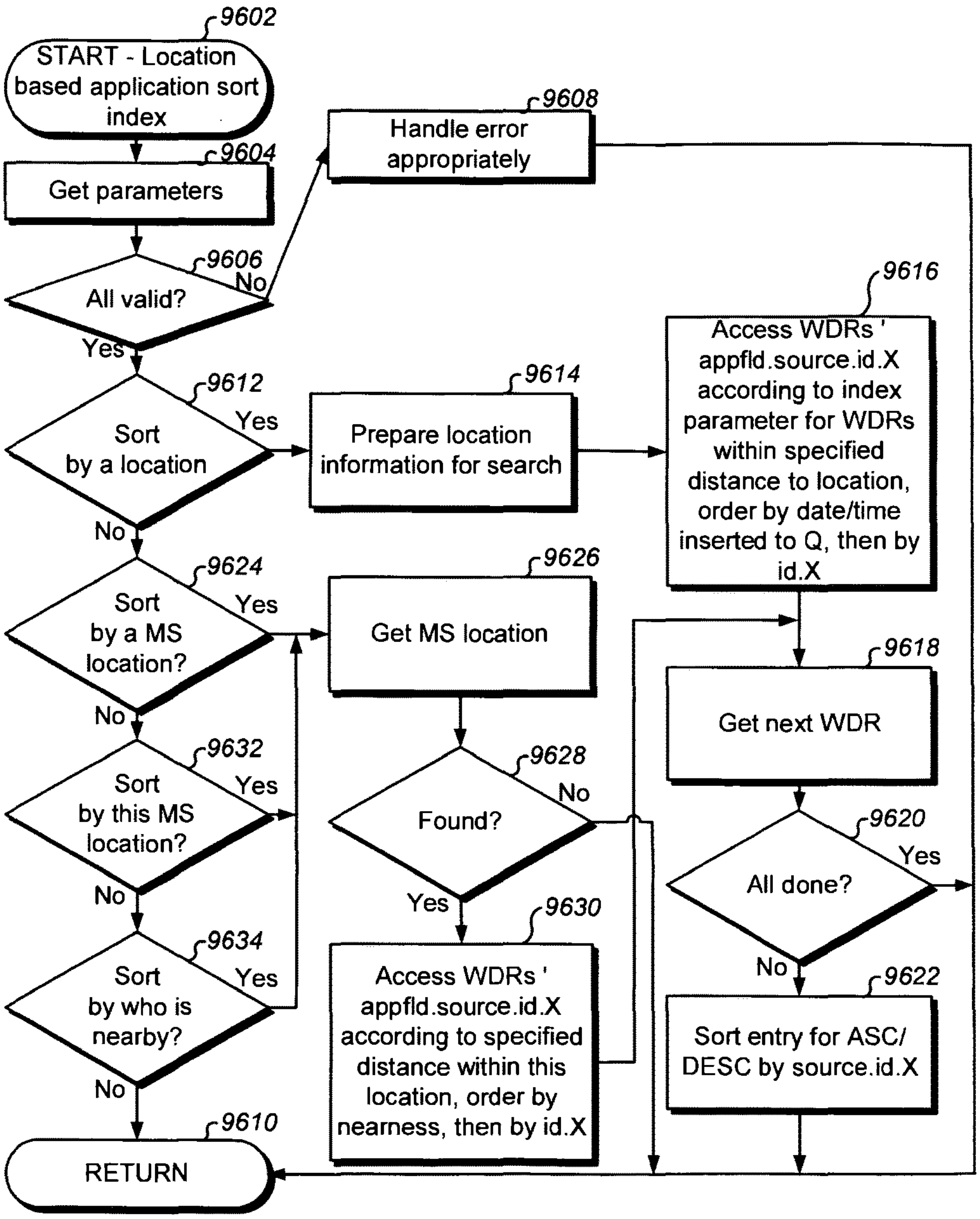
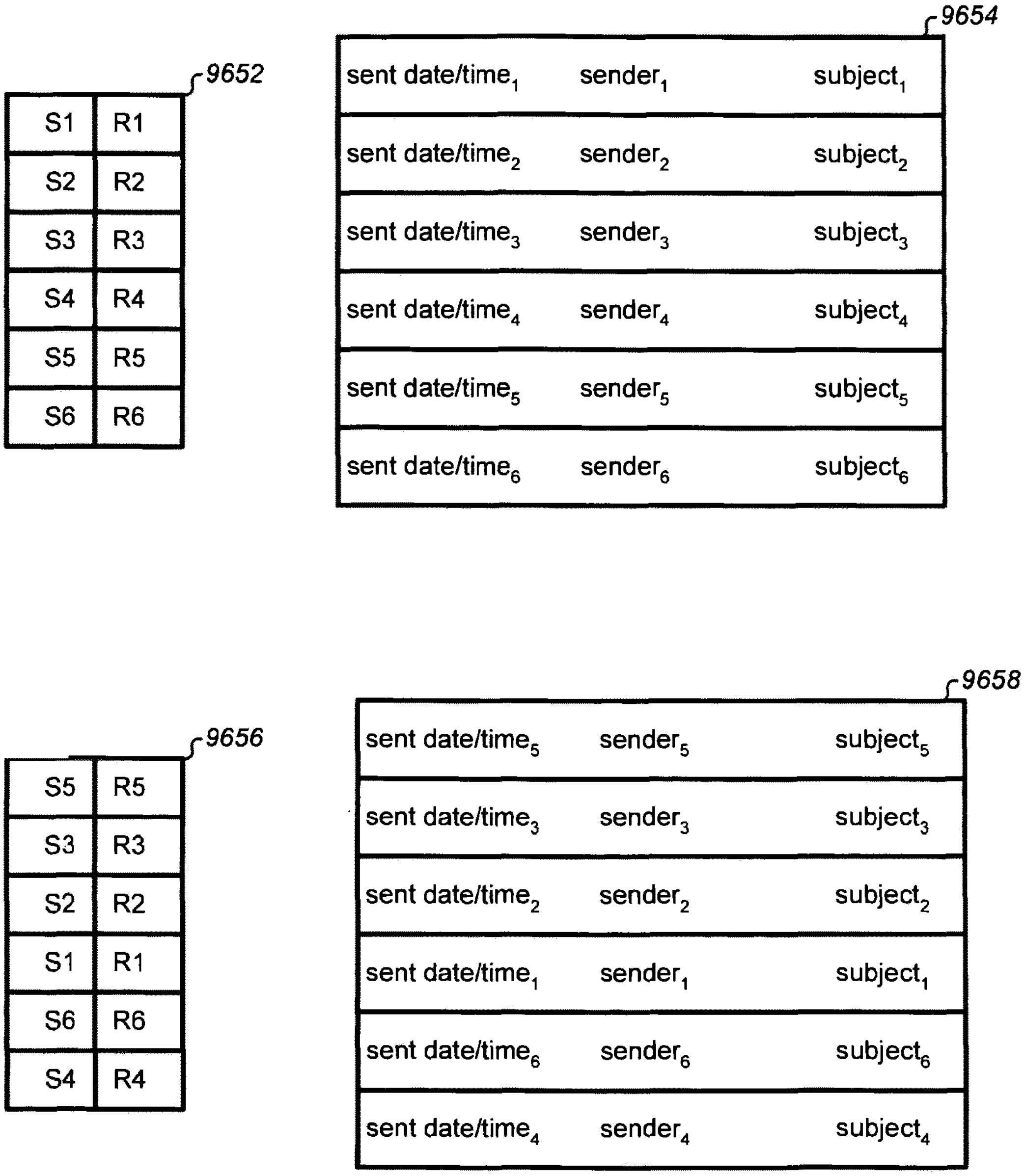

9602
START - Location based application sort index
9604
Get parameters
9606
All valid?
No
Yes
9608
Handle error appropriately
9612
Sort by a location
Yes
No
9614
Prepare location information for search
9616
Access WDRs ' appfld.source.id.X according to index parameter for WDRs within specified distance to location, order by date/time inserted to Q, then by id.X
9624
Sort by a MS location?
Yes
No
9626
Get MS location
9628
Found?
No
Yes
9630
Access WDRs ' appfld.source.id.X according to specified distance within this location, order by nearness, then by id.X
9618
Get next WDR
9620
All done?
Yes
No
9622
Sort entry for ASC/ DESC by source.id.X
9632
Sort by this MS location?
Yes
No
9634
Sort by who is nearby?
Yes
No
9610
RETURN

Fig. 96A



S1	R1
S2	R2
S3	R3
S4	R4
S5	R5
S6	R6

sent date/time$_1$	sender$_1$	subject$_1$
sent date/time$_2$	sender$_2$	subject$_2$
sent date/time$_3$	sender$_3$	subject$_3$
sent date/time$_4$	sender$_4$	subject$_4$
sent date/time$_5$	sender$_5$	subject$_5$
sent date/time$_6$	sender$_6$	subject$_6$

S5	R5
S3	R3
S2	R2
S1	R1
S6	R6
S4	R4

sent date/time$_5$	sender$_5$	subject$_5$
sent date/time$_3$	sender$_3$	subject$_3$
sent date/time$_2$	sender$_2$	subject$_2$
sent date/time$_1$	sender$_1$	subject$_1$
sent date/time$_6$	sender$_6$	subject$_6$
sent date/time$_4$	sender$_4$	subject$_4$

Fig. 96B

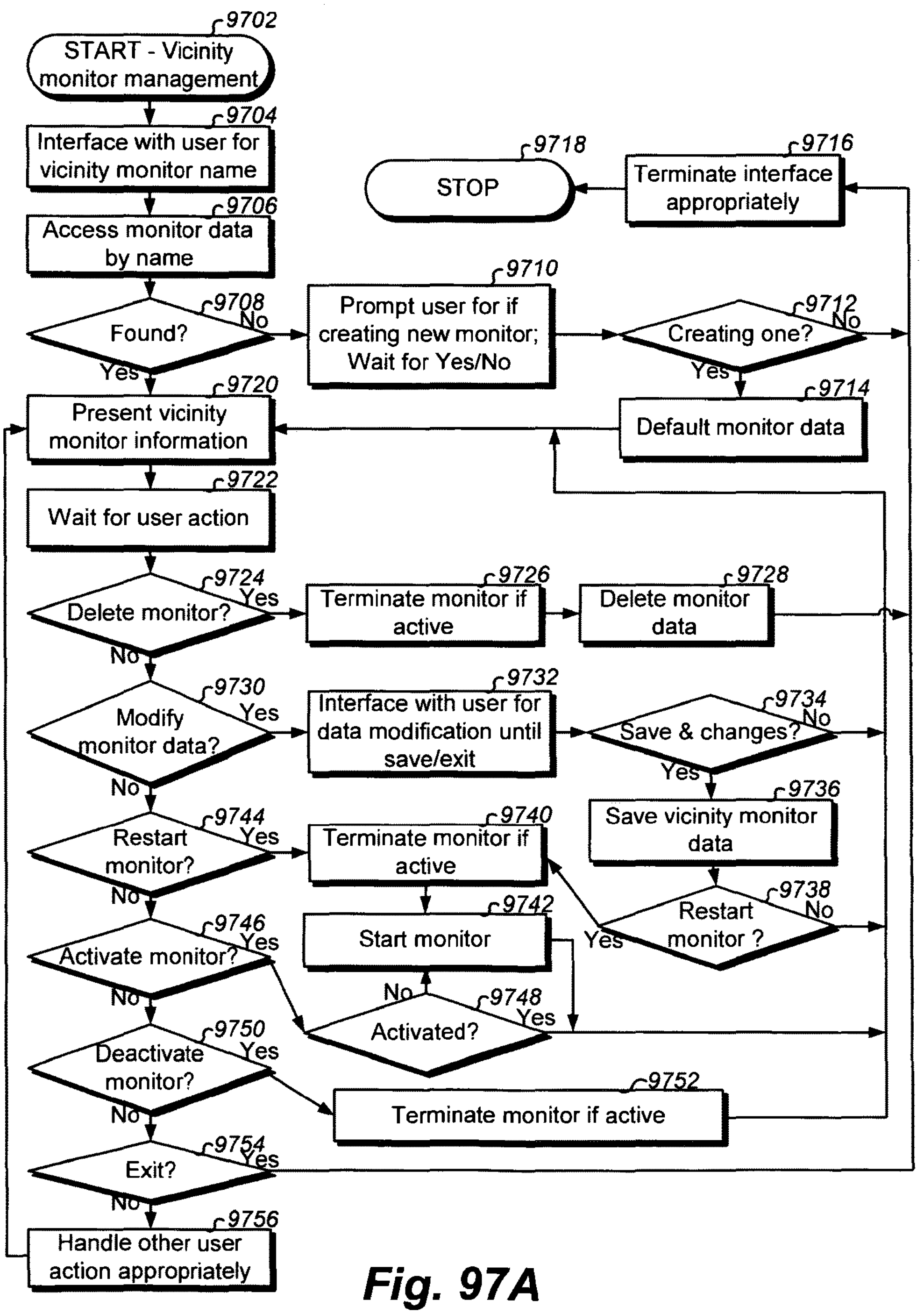
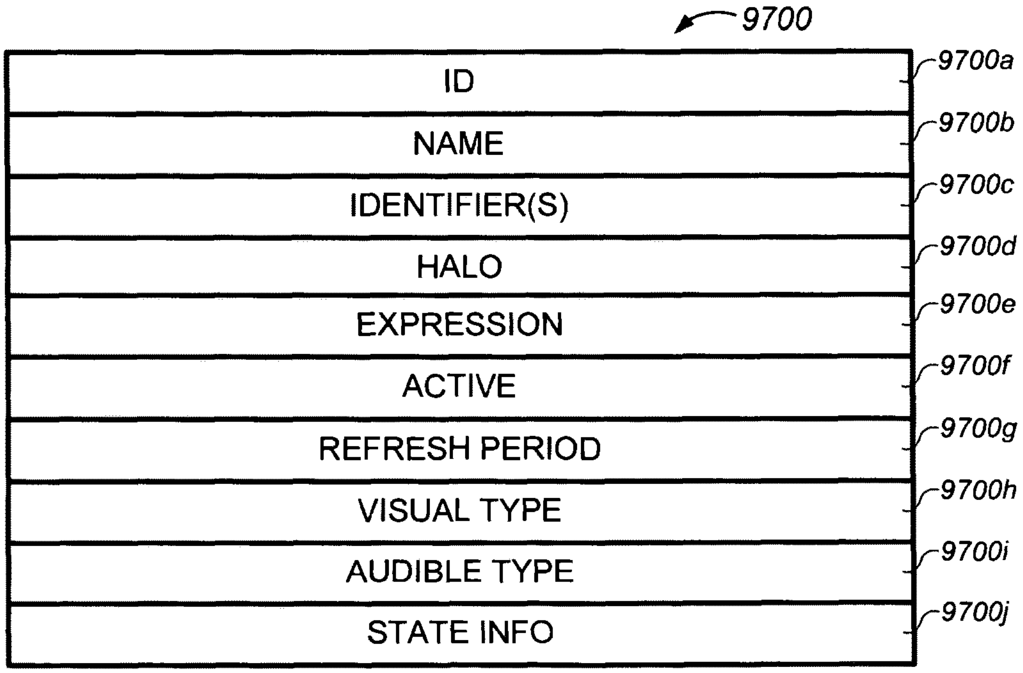

9702
START - Vicinity monitor management
9704
Interface with user for vicinity monitor name
9706
Access monitor data by name
9708
Found?
No
Yes
9710
Prompt user for if creating new monitor; Wait for Yes/No
9712
Creating one?
No
Yes
9714
Default monitor data
9716
Terminate interface appropriately
9718
STOP
9720
Present vicinity monitor information
9722
Wait for user action
9724
Delete monitor?
Yes
No
9726
Terminate monitor if active
9728
Delete monitor data
9730
Modify monitor data?
Yes
No
9732
Interface with user for data modification until save/exit
9734
Save & changes?
No
Yes
9736
Save vicinity monitor data
9738
Restart monitor ?
No
Yes
9744
Restart monitor?
Yes
No
9740
Terminate monitor if active
9742
Start monitor
9746
Activate monitor?
Yes
No
9748
Activated?
No
Yes
9750
Deactivate monitor?
Yes
No
9752
Terminate monitor if active
9754
Exit?
Yes
No
9756
Handle other user action appropriately

Fig. 97A



ID

NAME

IDENTIFIER(S)

HALO

EXPRESSION

ACTIVE

REFRESH PERIOD

VISUAL TYPE

AUDIBLE TYPE

STATE INFO

Fig. 97B

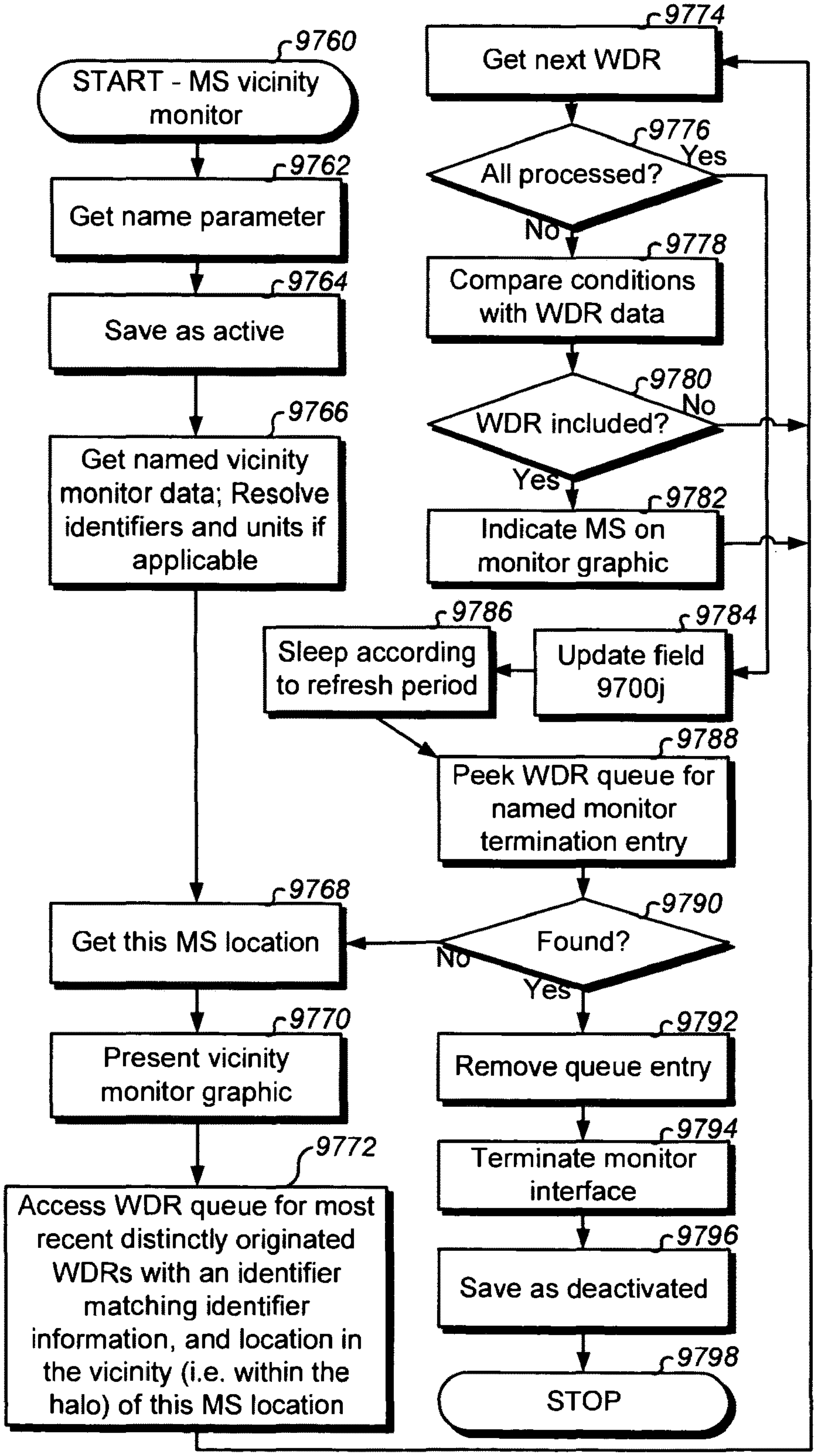

9760
START - MS vicinity monitor
9762
Get name parameter
9764
Save as active
9766
Get named vicinity monitor data; Resolve identifiers and units if applicable
9768
Get this MS location
9770
Present vicinity monitor graphic
9772
Access WDR queue for most recent distinctly originated WDRs with an identifier matching identifier information, and location in the vicinity (i.e. within the halo) of this MS location
9774
Get next WDR
9776
All processed?
Yes
No
9778
Compare conditions with WDR data
9780
WDR included?
No
Yes
9782
Indicate MS on monitor graphic
9784
Update field 9700j
9786
Sleep according to refresh period
9788
Peek WDR queue for named monitor termination entry
9790
Found?
No
Yes
9792
Remove queue entry
9794
Terminate monitor interface
9796
Save as deactivated
9798
STOP

Fig. 97C

SYSTEM AND METHOD FOR LOCATION BASED EXCHANGE NETWORK

CROSS-REFERENCES TO RELATED APPLICATIONS

This application is a continuation of application Ser. No. 15/218,039 filed Jul. 24, 2016 and entitled "System and Method for Sound Wave Triggered Content Delivery" which is a continuation of application Ser. No. 14/752,945 (now U.S. Pat. No. 9,456,303 issued on Sep. 27, 2016) filed Jun. 28, 2015 and entitled "System and Method for Service Access Via Hopped Wireless Mobile Device(s)" which is a continuation of application Ser. No. 13/972,125 (now U.S. Pat. No. 9,078,095 issued on Jul. 7, 2015) filed Aug. 21, 2013 and entitled "System and Method for Location Based Inventory Management" which is a continuation of application Ser. No. 12/590,831 (now U.S. Pat. No. 8,634,796 issued on Jan. 21, 2014) filed Nov. 13, 2009 and entitled "System and Method for Location Based Exchanges of Data Facilitating Distributed Locational Applications" which is a continuation in part of application Ser. No. 12/287,064 (now U.S. Pat. No. 8,639,267 issued on Jan. 28, 2014) filed Oct. 3, 2008 and entitled "System and Method for Location Based Exchanges of Data Facilitating Distributed Locational Applications" which is a continuation in part of application Ser. No. 12/077,041 (now U.S. Pat. No. 8,600, 341 issued on Dec. 3, 2013) filed Mar. 14, 2008 and entitled "System and Method for Location Based Exchanges of Data Facilitating Distributed Locational Applications". This application contains an identical specification to Ser. No. 15/218,039 except for the title, abstract, and claims.

FIELD OF THE INVENTION

The present disclosure relates generally to location based services for mobile data processing systems, and more particularly to location based exchanges of data between distributed mobile data processing systems for locational applications. A common connected service is not required for location based functionality and features. Location based exchanges of data between distributed mobile data processing systems enable location based features and functionality in a peer to peer manner.

BACKGROUND OF THE INVENTION

The internet has exploded with new service offerings. Websites yahoo.com, google.com, ebay.com, amazon.com, and iTunes.com have demonstrated well the ability to provide valuable services to a large dispersed geographic audience through the internet (ebay, yahoo, google, amazon and iTunes (Apple) are trademarks of the respective companies). Thousands of different types of web services are available for many kinds of functionality. Advantages of having a service as the intermediary point between clients, users, and systems, and their associated services, includes centralized processing, centralized maintaining of data, for example to have an all knowing database for scope of services provided, having a supervisory point of control, providing an administrator with access to data maintained by users of the web service, and other advantages associated with centralized control. The advantages are analogous to those provided by the traditional mainframe computer to its clients wherein the mainframe owns all resources, data, processing, and centralized control for all users and systems (clients) that access its services. However, as computers declined in price and

adequate processing power was brought to more distributed systems, such as Open Systems (i.e. Windows, UNIX, Linux, and Mac environments), the mainframe was no longer necessary for many of the daily computing tasks. In fact, adequate processing power is incorporated in highly mobile devices, various handheld mobile data processing systems, and other mobile data processing systems. Technology continues to drive improved processing power and data storage capabilities in less physical space of a device. Just as Open Systems took much of the load of computing off of mainframe computers, so to can mobile data processing systems offload tasks usually performed by connected web services. As mobile data processing systems are more capable, there is no need for a service to middleman interactions possible between them.

While a centralized service has its advantages, there are also disadvantages. A service becomes a clearinghouse for all web service transactions. Regardless of the number of threads of processing spread out over hardware and processor platforms, the web service itself can become a bottleneck causing poor performance for timely response, and can cause a large amount of data that must be kept for all connected users and/or systems. Even large web services mentioned above suffer from performance and maintenance overhead. A web service response will likely never be fast enough. Additionally, archives must be kept to ensure recovery in the event of a disaster because the service houses all data for its operations. Archives also require storage, processing power, planning, and maintenance. A significantly large and costly data center is necessary to accommodate millions of users and/or systems to connect to the service. There is a tremendous amount of overhead in providing such a service. Data center processing power, data capacity, data transmission bandwidth and speed, infrastructure entities, and various performance considerations are quite costly. Costs include real estate required, utility bills for electricity and cooling, system maintenance, personnel to operate a successful business with service(s), etc. A method is needed to prevent large data center costs while eliminating performance issues for features sought. It is inevitable that as users are hungry for more features and functionality on their mobile data processing systems, processing will be moved closer to the device for optimal performance and infrastructure cost savings.

Service delivered location dependent content was disclosed in U.S. Pat. Nos. 6,456,234; 6,731,238; 7,187,997 (Johnson). Anonymous location based services was disclosed in U.S. PTO Publication 2006/0022048 (Johnson). The Johnson patents and published application operate as most web services do in that the clients connecting to the service benefit from the service by having some connectivity to the service. U.S. Publication 2006/0022048 (Johnson) could cause large numbers of users to inundate the service with device heartbeats and data to maintain, depending on the configurations made. While this may be of little concern to a company that has successfully deployed substantially large web service resources, it may be of great concern to other more frugal companies. A method is needed for enabling location dependent features and functionality without the burden of requiring a service.

Users are skeptical about their privacy as internet services proliferate. A service by its very nature typically holds information for a user maintained in a centralized service database. The user's preferences, credential information, permissions, customizations, billing information, surfing habits, and other conceivable user configurations and activity monitoring, can be housed by the service at the service.

Company insiders, as well as outside attackers, may get access. Most people are concerned with preventing personal information of any type being kept in a centralized database which may potentially become compromised from a security standpoint. Location based services are of even more concern, in particular when the locations of the user are to be known to a centralized service. A method and system is needed for making users comfortable with knowing that their personal information is at less risk of being compromised.

A reasonable requirement is to push intelligence out to the mobile data processing systems themselves, for example, in knowing their own locations and perhaps the locations of other nearby mobile data processing systems. Mobile data processing systems can intelligently handle many of their own application requirements without depending on some remote service. Just as two people in a business organization should not need a manager to speak to each other, no two mobile data processing systems should require a service middleman for useful location dependent features and functionality. The knowing of its own location should not be the end of social interaction implementation local to the mobile data processing systems, but rather the starting place for a large number of useful distributed local applications that do not require a service.

Different users use different types of Mobile data processing Systems (MSs) which are also called mobile devices: laptops, tablet computers, Personal Computers (PCs), Personal Digital Assistants (PDAs), cell phones, automobile dashboard mounted data processing systems, shopping cart mounted data processing systems, mobile vehicle or apparatus mounted data processing systems, Personal Navigational Devices (PNDs), iPhones (iPhone is a trademark of Apple, Inc.), various handheld mobile data processing systems, etc. MSs move freely in the environment, and are unpredictably moveable (i.e. can be moved anywhere, anytime). Many of these Mobile data processing Systems (MSs) do not have capability of being automatically located, or are not using a service for being automatically located. Conventional methods use directly relative stationary references such as satellites, antennas, etc. to locate MSs. Stationary references are expensive to deploy, and risk obsolescence as new technologies are introduced to the marketplace. Stationary references have finite scope of support for locating MSs.

While the United States E911 mandate for cellular devices documents requirements for automatic location of a Mobile data processing System (MS) such as a cell phone, the mandate does not necessarily promote real time location and tracking of the MSs, nor does it define architecture for exploiting Location Based Services (LBS). We are in an era where Location Based Services (LBS), and location dependent features and functionality, are among the most promising technologies in the world. Automatic locating of every Mobile data processing System (MS) is an evolutionary trend. A method is needed to shorten the length of time for automatically locating every MS. Such a goal can be costly using prior art technologies such as GPS (Global Positioning System), radio wave triangulation, coming within range to a known located sensor, or the like. Complex system infrastructure, or added hardware costs to the MSs themselves, make such ventures costly and time constrained by schedules and costs involved in engineering, construction, and deployment.

A method is needed for enabling users to get location dependent features and functionality through having their mobile locations known, regardless of whether or not their MS is equipped for being located. Also, new and modern location dependent features and functionality can be provided to a MS unencumbered by a connected service.

BRIEF SUMMARY OF THE INVENTION

LBS (Location Based Services) is a term which has gained in popularity over the years as MSs incorporate various location capability. The word "Services" in that terminology plays a major role in location based features and functionality involving interaction between two or more users. This disclosure introduces a new terminology, system, and method referred to as Location Based eXchanges (LBX). LBX is an acronym used interchangeably/contextually throughout this disclosure for the singular term "Location Based Exchange" and for the plural term "Location Based Exchanges", much the same way LBS is used interchangeably/contextually for the single term "Location Based Service" and for the plural term "Location Based Services". LBX describes leveraging the distributed nature of connectivity between MSs in lieu of leveraging a common centralized service nature of connectivity between MSs. The line can become blurred between LBS and LBX since the same or similar features and functionality are provided, and in some cases strengths from both may be used. The underlying architectural shift differentiates LBX from LBS for depending less on centralized services, and more on distributed interactions between MSs. LBX provide server-free and server-less location dependent features and functionality.

Disclosed are many different aspects to LBX, starting with the foundation requirement for each participating MS to know, at some point in time, their own whereabouts. LBX is enabled when an MS knows its own whereabouts. It is therefore a goal to first make as many MSs know their own whereabouts as possible. When two or more MSs know their own whereabouts, LBX enables distributed locational applications whereby a server is not required to middleman social interactions between the MSs. The MSs interact as peers. LBX disclosed include purely peer to peer interactions, peer to peer interactions for routing services, peer to peer interactions for delivering distributed services, and peer to peer interactions for location dependent features and functionality (e.g. a first mobile data processing system sends directly (e.g. wirelessly) to a second mobile data processing system without using an intervening data processing system). One embodiment of an LBX enabled MS is referred to as an lbxPhone™.

It is an advantage herein to have no centralized service governing location based features and functionality among MSs. Avoiding a centralized service prevents performance issues, infrastructure costs, and solves many of the issues described above. No centralized service also prevents a user's information from being kept in one accessible place. LBS contain centralized data that is personal in nature to its users. This is a security concern. Having information for all users in one place increases the likelihood that a disaster to the data will affect more than a single user. LBX spreads data out across participating systems so that a disaster affecting one user does not affect any other user.

It is an advantage herein for enabling useful distributed applications without the necessity of having a service, and without the necessity of users and/or systems registering with a service. MSs interact as peers in preferred embodiments, rather than as clients to a common service (e.g. internet connected web service).

It is an advantage herein for locating as many MSs as possible in a wireless network, and without additional deployment costs on the MSs or the network. Conventional locating capability includes GPS (Global Positioning System) using stationary orbiting satellites, improved forms of GPS, for example AGPS (Adjusted GPS) and DGPS (Differential GPS) using stationary located ground stations, wireless communications to stationary located cell tower base stations, TDOA (Time Difference of Arrival) or AOA (Angle of Arrival) triangulation using stationary located antennas, presence detection in vicinity of a stationary located antenna, presence detection at a wired connectivity stationary network location, or other conventional locating systems and methods. Mobile data processing systems, referred to as Indirectly Located Mobile data processing systems (ILMs), are automatically located using automatically detected locations of Directly Located Mobile data processing systems (DLMs) and/or automatically detected locations of other ILMs. ILMs are provided with the ability to participate in the same LBS, or LBX, as a DLM (Directly Located Mobile data processing system). DLMs are located using conventional locating capability mentioned above. DLMs provide reference locations for automatically locating ILMs, regardless of where any one is currently located. DLMs and ILMs can be highly mobile, for example when in use by a user. There are a variety of novel methods for automatically locating ILMs, for example triangulating an ILM (Indirectly Located Mobile data processing system) location using a plurality of DLMs, detecting the ILM being within the vicinity of at least one DLM, triangulating an ILM location using a plurality of other ILMs, detecting the ILM being within the vicinity of at least one other ILM, triangulating an ILM location using a mixed set of DLM(s) and ILM(s), determining the ILM location from heterogeneously located DLMs and/or ILMs, and other novel methods.

MSs are automatically located without using direct conventional means for being automatically located. The conventional locating capability (i.e. conventional locating methods) described above is also referred to as direct methods. Conventional methods are direct methods, but not all direct methods are conventional. There are new direct techniques disclosed below. Provided herein is an architecture, as well as systems and methods, for immediately bringing automatic location detection to every MS in the world, regardless of whether that MS is equipped for being directly located. MSs without capability of being directly located are located by leveraging the automatically detected locations of MSs that are directly located. This is referred to as being indirectly located. An MS which is directly located is hereinafter referred to as a Directly Located Mobile data processing system (DLM). For a plural acronym, MSs which are directly located are hereinafter referred to as Directly Located Mobile data processing systems (DLMs). MSs without capability of being directly located are located using the automatically detected locations of MSs that have already been located. An MS which is indirectly located is hereinafter referred to as an Indirectly Located Mobile data processing system (ILM). For a plural acronym, MSs which are indirectly located are hereinafter referred to as Indirectly Located Mobile data processing systems (ILMs). A DLM can be located in the following ways:

A) New triangulated wave forms;

B) Missing Part Triangulation (MPT) as disclosed below;

C) Heterogeneous direct locating methods;

D) Assisted Direct Location Technology (ADLT) using a combination of direct and indirect methods;

E) Manually specified; and/or

F) Any combinations of A) through E);

DLMs provide reference locations for automatically locating ILMs, regardless of where the DLMs are currently located. It is preferable to assure an accurate location of every DLM, or at least provide a confidence value of the accuracy. A confidence value of the accuracy is used by relative ILMs to determine which are the best set (e.g. which are of highest priority for use to determine ILM whereabouts) of relative DLMs (and/or ILMs) to use for automatically determining the location of the ILM.

In one example, the mobile locations of several MSs are automatically detected using their local GPS chips. Each is referred to as a DLM. The mobile location of a non-locatable MS is triangulated using radio waves between it and three (3) of the GPS equipped DLMs. The MS becomes an ILM upon having its location determined relative the DLMs. ILMs are automatically located using DLMs, or other already located ILMs. An ILM can be located in the following ways:

G) Triangulating an ILM location using a plurality of DLMs with wave forms of any variety (e.g. AOA, TDOA, MPT (a heterogeneous location method));

H) Detecting the ILM being within the reasonably close vicinity of at least one DLM;

I) Triangulating an ILM location using a plurality of other ILMs with wave forms of any variety;

J) Detecting the ILM being within the reasonable close vicinity of at least one other ILM;

K) Triangulating an ILM location using a mixed set of DLM(s) and ILM(s) with wave forms of any variety (referred to as ADLT);

L) Determining the ILM location from heterogeneously located DLMs and/or ILMs (i.e. heterogeneously located, as used here, implies having been located relative different location methodologies);

M) A) through F) Above; and/or

N) Any combinations of A) through M).

Locating functionality may leverage GPS functionality, including but not limited to GPS, AGPS (Adjusted GPS), DGPS, (Differential GPS), or any improved GPS embodiment to achieve higher accuracy using known locations, for example ground based reference locations. The NexTel GPS enabled iSeries cell phones provide excellent examples for use as DLMs (Nextel is a trademark of Sprint/Nextel). Locating functionality may incorporate triangulated locating of the MS, for example using a class of Radio Frequency (RF) wave spectrum (cellular, WiFi (some WiFi embodiments referred to as WiMax), bluetooth, etc), and may use measurements from different wave spectrums for a single location determination (depends on communications interface(s) **70** available). A MS may have its whereabouts determined using a plurality of wave spectrum classes available to it (cellular, WiFi, bluetooth, etc). The term “WiFi” used throughout this disclosure also refers to the industry term “WiMax”. Locating functionality may include in-range proximity detection for detecting the presence of the MS. Wave forms for triangulated locating also include microwaves, infrared wave spectrum relative infrared sensors, visible light wave spectrum relative light visible light wave sensors, ultraviolet wave spectrum relative ultraviolet wave sensors, X-ray wave spectrum relative X-ray wave sensors, gamma ray wave spectrum relative gamma ray wave sensors, and longwave spectrum (below AM) relative longwave sensors. While there are certainly more common methods for automatically locating a MS (e.g. radio wave triangulation, GPS, in range proximity detection), those skilled in the art recognize there are methods for different wave spectrums being detected, measured, and used for carrying information between data processing systems.

Kubler et al (U.S. PTO publications 2004/0264442, 2004/0246940, 2004/0228330, 2004/0151151) disclosed methods for detecting presence of mobile entities as they come within range of a sensor. In Kubler et al, accuracy of the location of the detected MS is not well known, so an estimated area of the whereabouts of the MS is enough to accomplish intended functionality, for example in warehouse installations. A confidence value of this disclosure associated with Kubler et al tends to be low (i.e. not confident), with lower values for long range sensors and higher values for short range sensors.

GPS and the abundance of methods for improving GPS accuracy has led to many successful systems for located MSs with high accuracy. Triangulation provides high accuracies for locating MSs. A confidence value of this disclosure associated with GPS and triangulating location methods tends to be high (i.e. confident). It is preferred that DLMs use the highest possible accuracy method available so that relative ILMs are well located. Not all DLMs need to use the same location methods. An ILM can be located relative DLMs, or other ILMs, that each has different locating methodologies utilized.

Another advantage herein is to generically locate MSs using varieties and combinations of different technologies. MSs can be automatically located using direct conventional methods for accuracy to base on the locating of other MSs. MSs can be automatically located using indirect methods. Further, it is an advantage to indirectly locate a MS relative heterogeneously located MSs. For example, one DLM may be automatically located using GPS. Another DLM may be automatically located using cell tower triangulation. A third DLM may be automatically located using within range proximity. An ILM can be automatically located at a single location, or different locations over time, relative these three differently located DLMs. The automatically detected location of the ILM may be determined using a form of triangulation relative the three DLMs just discussed, even though each DLM had a different direct location method used. In a preferred embodiment, industry standard IEEE 802.11 WiFi is used to locate (triangulate) an ILM relative a plurality of DLMs (e.g. TDOA in one embodiment). This standard is prolific among more compute trended MSs. Any of the family of 802.11 wave forms such as 802.11a, 802.11b, 802.11g, or any other similar class of wave spectrum can be used, and the same spectrum need not be used between a single ILM and multiple DLMs. 802.x used herein generally refers to the many **802**. whatever variations.

Another advantage herein is to make use of existing marketplace communications hardware, communications software interfaces, and communications methods and location methods where possible to accomplish locating an MS relative one or more other MSs. While 802.x is widespread for WiFi communications, other RF wave forms can be used (e.g. cell phone to cell tower communications). In fact, any wave spectrum for carrying data applies herein. Of course, any protocol(s) may be involved in embodiments of the disclosures (e.g. TDMA, CDMA, H.323, SIP, 2G, 3G, ip phone, digital, analog, spectrum frequency, etc).

Still another advantage is for support of heterogeneous locatable devices. Different people like different types of devices as described above. Complete automation of locating functionality can be provided to a device through local automatic location detection means, or by automatic location detection means remote to the device. Also, an ILM can be located relative a laptop, a cell phone, and a PDA (i.e. different device types).

Yet another advantage is to prevent the unnecessary storing of large amounts of positioning data for a network of MSs. Keeping positioning data for knowing the whereabouts of all devices can be expensive in terms of storage, infrastructure, performance, backup, and disaster recovery. A preferred embodiment simply uses a distributed approach to determining locations of MSs without the overhead of an all-knowing database maintained somewhere. Positions of MSs can be determined "on the fly" without storing information in a master database. However, there are embodiments for storing a master database, or a subset thereof, to configurable storage destinations, when it makes sense. A subset can be stored at a MS.

Another advantage includes making use of existing location equipped MSs to expand the network of locatable devices by locating non-equipped MSs relative the location of equipped MSs. MSs themselves help increase dimensions of the locatable network of MSs. The locatable network of MSs is referred to as an LN-Expanse (i.e. Location-Network Expanse). An LN-Expanse dynamically grows and shrinks based on where MSs are located at a particular time. For example, as users travel with their personal MSs, the personal MSs themselves define the LN-Expanse since the personal MSs are used to locate other MSs. An ILM simply needs location awareness relative located MSs (DLMs and/or ILMs).

Yet another advantage is a MS interchangeably taking on the role of a DLM or ILM as it travels. MSs are chameleons in this regard, in response to location technologies that happen to be available. A MS may be equipped for DLM capability, but may be in a location at some time where the capability is inoperable. In these situations the DLM takes on the role of an ILM. When the MS again enters a location where it can be a DLM, it automatically takes on the role of the DLM. This is very important, in particular for emergency situations. A hiker has a serious accident in the mountains which prevents GPS equipped DLM capability from working. Fortunately, the MS automatically takes on the role of an ILM and is located within the vicinity of neighboring (nearby) MSs. This allows the hiker to communicate his location, operate useful locational application functions and features at his MS, and enable emergency help that can find him.

It is a further advantage that MS locations be triangulated using any wave forms (e.g. RF, microwaves, infrared, visible light, ultraviolet, X-ray, gamma ray). X-ray and gamma ray applications are special in that such waves are harmful to humans in short periods of times, and such applications should be well warranted to use such wave forms. In some medical embodiments, micro-machines may be deployed within a human body. Such micro-machines can be equipped as MSs. Wave spectrums available at the time of deployment can be used by the MSs for determining exact positions when traveling to through a body.

It is another advantage to use TDOA (Time Difference Of Arrival), AOA (Angle Of Arrival), and Missing Part Triangulation (MPT) when locating a MS. TDOA uses time information to determine locations, for example for distances of sides of a triangle. AOA uses angles of arrival to antennas to geometrically assess where a MS is located by intersecting lines drawn from the antennas with detected angles. MPT is disclosed herein as using combinations of AOA and TDOA to determine a location. Exclusively using all AOA or exclusively using all TDOA is not necessary. MPT can be a direct method for locating MSs.

Yet another advantage is to locate MSs using Assisted Direct Location Technology (ADLT). ADLT is disclosed herein as using direct (conventional) location capability together with indirect location capability to confidently determine the location of a MS.

Still another advantage is to permit manual specification for identifying the location of a MS (a DLM). The manual location can then in turn be used to facilitate locating other MSs. A user interface may be used for specification of a DLM location. The user interface can be local, or remote, to the DLM. Various manual specification methods are disclosed. Manual specification is preferably used with less mobile MSs, or existing MSs such as those that use dodgeball.com (trademark of Google). The confidence value depends on how the location is specified, whether or not it was validated, and how it changes when the MS moves after being manually set. Manual specification should have limited scope in an LN-expanse unless inaccuracies can be avoided.

Another advantage herein is locating a MS using any of the methodologies above, any combinations of the methodologies above, and any combinations of direct and/or indirect location methods described.

Another advantage is providing synergy between different locating technologies for smooth operations as an MS travels. There are large numbers of methods and combinations of those methods for keeping an MS informed of its whereabouts. Keeping an MS informed of its whereabouts in a timely manner is critical in ensuring LBX operate optimally, and for ensuring nearby MSs without certain locating technologies can in turn be located.

It is another advantage for locating an MS with multiple location technologies during its travels, and in using the best of breed data from multiple location technologies to infer a MS location confidently. Confidence values are associated with reference location information to ensure an MS using the location information can assess accuracy. A DLM is usually an "affirmifier". An affirmifier is an MS with its whereabouts information having high confidence of accuracy and can serve as a reference for other MSs. An ILM can also be an affirmifier provided there is high confidence that the ILM location is known. An MS (e.g. ILM) may be a "pacifier". A pacifier is an MS having location information for its whereabouts with a low confidence for accuracy. While it can serve as a reference to other ILMs, it can only do so by contributing a low confidence of accuracy.

It is another advantage for providing user customization of confidence values based on the user's experience. A MS user may completely rely on the MS system settings for setting confidence values, or may "tweak" location technology confidence values to accommodate experiences with particular location technologies that have been encountered during travels.

It is an advantage to synergistically make use of the large number of locating technologies available to prevent one particular type of technology to dominate others while using the best features of each to assess accurate mobile locations of MSs.

A further advantage is to leverage a data processing system with capability of being located for co-locating another data processing system without any capability of being located. For example, a driver owns an older model automobile, has a useful second data processing system in the automobile without means for being automatically located. The driver also own a cell phone, called a first data

processing system, which does have means for being automatically located. The location of the first data processing system can be shared with the second data processing system for locating the second data processing system. Further still, the second data processing system without means for being automatically located is located relative a first set (plurality) of data processing systems which are not at the same location as the second data processing system. So, data processing systems are automatically located relative at least one other data processing which can be automatically located.

Another advantage is a LBX enabled MS includes a service informant component for keeping a supervisory service informed. This prevents an MS from operating in total isolation, and prevents an MS from operating in isolation with those MSs that are within its vicinity (e.g. within maximum range **1306**) at some point in time, but to also participate when the same MSs are great distances from each other. There are LBX which would fit well into an LBS model, but a preferred embodiment chooses to use the LBX model. For example, multiple MS users are seeking to carpool to and from a common destination. The service informant component can perform timely updates to a supervisory service for route comparisons between MSs, even though periods of information are maintained only at the MSs. For example, users find out that they go to the same church with similar schedules, or coworkers find out they live nearby and have identical work schedules. The service informant component can keep a service informed of MS whereabouts to facilitate novel LBX applications. The service informant can also be configured for: communicating directly to another MS, communicating to a data processing system through a propagate-able service, invoking a "plug-in" home grown interface, alerting the MS user with a specified alert, or invoking an atomic command used by charter processing.

It is a further advantage in leveraging the vast amount of MS WiFi/WiMax deployment underway in the United States. More widespread WiFi/WiMax availability enhances the ability for well performing peer to peer types of features and functionality disclosed.

It is a further advantage to prevent unnecessary established connections from interfering with successfully triangulating a MS position. As the MS roams and encounters various wave spectrum signals, that is all that is required for determining the MS location. Broadcast signaling contains the necessary location information for automatically locating the MS.

Yet another advantage is to leverage Network Time Protocol (NTP) for eliminating bidirectional communications in determining Time of Arrival (TOA) and TDOA (Time Difference Of Arrival) measurements (TDOA as used in the disclosure generally refers to both TOA and TDOA). NTP enables a single unidirectional transmission of data to carry all that is necessary in determining TDOA, provided the sending data processing system and the receiving data processing system are NTP synchronized to an adequate granulation of time.

A further advantage is for making available to remote peer MSs certain MS operating system resources such as memory, storage, semaphores, application data, or to the like, according to permissions. A single MS can access and use operating system resources of another MS, for example in charter processing. Also, semaphore controlled synchronization of processing can be achieved over a network, or plurality, of peer MSs without a common server to synchronize the processing.

It is an advantage of this disclosure to provide a competing superior alternative to server based mobile technologies such as that of U.S. Pat. Nos. 6,456,234; 6,731,238; 7,187,997; and U.S. PTO Publication 2006/0022048 (Johnson). It is also an advantage to leverage both LBX technology and LBS technology in the same MS in order to improve the user experience. The different technologies can be used to complement each other in certain embodiments.

A further advantage herein is to leverage existing "usual communications" data transmissions for carrying new data that is ignored by existing MS processing, but observed by new MS processing, for carrying out processing maximizing location functions and features across a large geography. Alternatively, new data can be transmitted between systems for the same functionality.

It is an advantage herein in providing peer to peer service propagation. ILMs are provided with the ability to participate in the same Location Based Services (LBS) or other services as DLM(s) in the vicinity. An MS may have access to services which are unavailable to other MSs. Any MS can share its accessible services for being accessible to any other MS, preferably in accordance with permissions. For example, an MS without internet access can get internet access via an MS in the vicinity with internet access. In a preferred embodiment, permissions are maintained in a peer to peer manner prior to lookup for proper service sharing. In another embodiment, permissions are specified and used at the time of granting access to the shared services. Once granted for sharing, services can be used in a mode as if the sharing user is using the services, or in a mode as if the user accepting the share is a new user to the service. Routing paths are dynamically reconfigured and transparently used as MSs travel. Hop counts dynamically change to strive for a minimal number of hops for an MS getting access to a desirable service. Route communications depend on where the MS needing the service is located relative a minimal number of hops through other MSs to get to the service. Services can be propagated from DLMs to DLMs, DLMs to ILMs, ILMs to DLMs, or ILMs to ILMs.

Services otherwise unavailable to a first MS (or MS user) in the LN-Expanse become available through another MS which does have access to the service. A plurality of MSs may facilitate the connection (e.g. hops) from the first MS to the last MS which publishes the service and has access to the service. MSs can access needed services through MSs in the vicinity when necessary. A service directory is shared and propagated between MSs so that the superset of services in a LN-Expanse are made available to any one MS in the LN-Expanse regardless of current MS conditions, whereabouts, capability, or an inability to connect to a desired service. A service route is minimized for best performance even with highly mobile MSs by minimizing a number of hops between MSs to reach a service.

It is another advantage herein for providing peer to peer permissions, authentication, and access control. A service is not necessary for maintaining credentials and permissions between MSs. Permissions are maintained locally to a MS. In a centralized services model, a database can become massive in size when searching for needed permissions. Permission searching and validation of U.S. PTO Publication 2006/0022048 (Johnson) was costly in terms of database size and performance. There was overhead in maintaining who owned the permission configuration for every permission granted. Maintaining permissions locally, as described below, reduces the amount of data to represent the permission because the owner is understood to be the personal user of the MS. Additionally, permission searching is very fast because the MS only has to search its local data for permissions that apply to only its MS.

Yet another advantage is to provide a nearby, or nearness, status using a peer to peer system and method, rather than intelligence maintained in a centralized database for all participating MSs. There is lots of overhead in maintaining a large database containing locations of all known MSs. This disclosure removes such overhead through using nearby detection means of one MS when in the vicinity of another MS. There are varieties of controls for governing how to generate the nearby status. In one aspect, a MS automatically calls the nearby MS thereby automatically connecting the parties to a conversation without user interaction to initiate the call. In another aspect, locally maintained configurations govern functionality when MSs are newly nearby, or are newly departing being nearby. Nearby status, alerts, and queries are achieved in a LBX manner.

It is yet another advantage for automatic call forwarding, call handling, and call processing based on the whereabouts of a MS, or whereabouts of a MS relative other MSs. The nearness condition of one MS to another MS can also affect the automatic call forwarding functionality.

Yet another advantage herein is for peer to peer content delivery and local MS configuration of that content. Users need no connectivity to a service. Users make local configurations to enjoy location based content delivery to other MSs. Content is delivered under a variety of circumstances for a variety of configurable reasons. Content maintained local to an MS is delivered asynchronously to other MSs for nearby alerts, arrival or departure to and from geofenced areas, and other predicated conditions of nearby MSs. While it may appear there are LBS made available to users of MSs, there are in fact LBX being made available to those users.

Another advantage herein is a LBX enabled MS can operate in a peer to peer manner to data processing systems which control environmental conditions. For example, automobile equipped (or driver kept) MSs encounter an intersection having a traffic light. Interactions between the MSs at the intersection and a data processing system in the vicinity for controlling the traffic light can automatically override light color changing for optimal traffic flow. In another embodiment, a parking lot search by a user with an MS is facilitated as he enters the parking lot, and in accordance with parking spaces currently occupied. In general, other nearby data processing systems can have their control logic processed for a user's preferences (as defined in the MS), a group of nearby user's preferences, and/or situational locations (see U.S. Pat. Nos. 6,456,234; 6,731,238; 7,187,997 (Johnson) for "situational location" terminology) of nearby MSs.

Another advantage herein is an MS maintains history of hotspot locations detected for providing graphical indication of hotspot whereabouts. This information can be used by the MS user in guiding where a user should travel in the future for access to services at the hotspot. Hotspot growth prevents a database in being timely configured with new locations. The MS can learn where hotspots are located, as relevant to the particular MS. The hotspot information is instantly available to the MS.

A further advantage is for peer to peer proximity detection for identifying a peer service target within the MS vicinity. A peer service target can be acted upon by an MS within range, using an application at the MS. The complementary whereabouts of the peer service target and MS automatically notify the user of service availability. The user can then use the MS application for making a payment, or for performing an account transfer, account deposit, account deduction, or any other transaction associated with the peer service target.

Yet another advantage is for a MS to provide new self management capability such as automatically marking photographs taken with location information, a date/time stamp, and who was with the person taking the picture.

Yet another advantage is being alerted to nearby people needing assistance and nearby fire engines or police cars that need access to roads.

A further advantage is providing a MS platform for which new LBX features and functionality can be brought quickly to the marketplace. The platform caters to a full spectrum of users including highly technical software developers, novice users, and users between those ranges. A rich programming environment is provided wherein whereabouts (WDR) information interchanged with other MSs in the vicinity causes triggering of privileged actions configured by users. The programming environment can be embedded in, or "plugged into", an existing software development environment, or provided on its own. A syntax may be specified with source code statements, XML, SQL database definitions, a datastream, or any other derivative of a well defined BNF grammar. A user friendly configuration environment is provided wherein whereabouts information interchanged with other MSs in the vicinity causes triggering of privileged actions configured by users. The platform is an event based environment wherein WDRs containing certain configured sought information are recognized at strategic processing paths for causing novel processing of actions. Events can be defined with complex expressions, and actions can be defined using homegrown executables, APIs, scripts, applications, a set of commands provided with the LBX platform, or any other executable processing. The LBX platform includes a variety of embodiments for charter and permission definitions including an internalized programmatic form, a SQL database form, a data record form, a datastream form, and a well defined BNF grammar for deriving other useful implementations (e.g. lex and yacc).

It is an advantage for permissions and/or charters to be configured in anticipation of every possible future travel, situation, environment, application, or condition of a MS (or MS user), or a plurality of related (by permissions and charters) MSs (or MS users). It is powerful in how permissions and charters configured in advance of anticipated events reveal novel unpredictably timed automated actions and application behavior for novel uses.

It is another advantage to support a countless number of privileges that can be configured, managed, and processed in a peer to peer manner between MSs. Any peer to peer feature or set of functionality can have a privilege associated to it for being granted from one user to another. It is also an advantage for providing a variety of embodiments for how to manage and maintain privileges in a network of MSs.

It is another advantage to support a complete set of options for charters that can be configured, managed, and processed in a peer to peer manner between MSs. Charters can become effective under a comprehensive set of conditions, expressions, terms, and operators. It is also an advantage for providing a variety of embodiments for how to manage and maintain charters in a network of MSs. Charters themselves can be self modifying for changing permissions or charters "on the fly" (i.e. during charter processing).

It is a further advantage for providing multithreaded communications of permission and charter information and transactions between MSs for well performing peer to peer interactions. Any signal spectrum for carrying out transmission and reception is candidate, depending on the variety of MS. In fact, different signaling wave spectrums, types, and protocols may be used in interoperating communications, or even for a single transaction, between MSs.

It is yet another advantage for increasing the range of the LN-expanse from a wireless vicinity to potentially infinite vicinity through other data processing (e.g. routing) equipment. While wireless proximity is used for governing automatic location determination, whereabouts information may be communicated between MSs great distances from each other provided there are privileges and/or charters in place making such whereabouts information relevant for the MS. Whereabouts information of others will not be maintained unless there are privileges in place to maintain it. Whereabouts information may not be shared with others if there have been no privileges granted to a potential receiving MS. Privileges can provide relevance to what whereabouts (WDR) information is of use, or should be processed, maintained, or acted upon.

Another advantage is to provide a MS which can be user configured for any desired behavior based on location, whereabouts, and "in the vicinity" conditions for the MS and/or its peer MSs during travels. A user has infinite control over providing a processing "character" for the MS. Also, various MS applications are generically supported with integrated locational based features and functionality. Charters may be used to automatically perform: MS configuration and system variable setting, clip-board and paste operations, MS input and output control, automatic communications with other MSs or data processing systems, enabling/disabling a feature or service, and many other features.

Another advantage is for using a convenient user interface such as map navigation for generating a map term such as a point, point and radius, or set of points defining area(s) on a map which is conveniently referenced in a charter configuration and later processed for replacement. For example, a user makes selection(s) on a map, and location information is automatically generated for the selection(s). The user can assign a convenient name to the location information without knowing details of the location information itself. The user can then reference the name for completely specifying the associated location information details. Also, the user may use WDR search criteria for determining a map term, the WDR found being one originated from the MS of map term creation or that of a peer MS. Recent whereabouts of a WDR found (e.g. from queue **22**), or past whereabouts of a WDR found (e.g. history **30**) may be used. Queue **22** may be viewed as maintaining a short term history, while history **30** may be viewed as maintaining a longer term history. Specifying locations in charter configurations can be tedious. Map terms provide the user with a simple user interface method to specify locations, and for hiding complexities of how the location was determined and generated for charter use. In some embodiments, map terms are used in broader scope by permitting any substitution where referenced. In some embodiments, map terms are used in broader scope by permitting "special terms" to be automatically created by a user by simply selecting a MS on a map.

It is an advantage for a convenient "charters starters" user interface for browsing, enabling, disabling, and maintaining charters depending on application, categories, or useable/clone-able snippets of the charters. For example, a MS may come prepackaged with many charters which have been organized and marked for particular applications and categories. The user can search, find, manage and enable/disable a set of charters based on their application or category, and can clone charter subsets for creating new charters. A MS user may manage his own charters, or charters of privilege granting others, using the charters starters interfaces. The user is also able to search, find, manage and enable/disable a set of charters based on any criteria found in the charter definitions themselves. A knowledgeable or authorized user may organize charters as he sees fit, for example to assign charters to categories and applications. The charter starters user interface organizes charters in easily identifiable groups (e.g. folders, categories, applications, etc) and provides simplicity for enabling, disabling and organizing any desired sets of complex charter configurations.

It is an advantage in providing application term triggered processing to the LBX platform described, and for all users and skill sets thereof. A rich programming environment and user friendly configuration environment is provided wherein application data which becomes modified causes triggering of privileged actions configured by users. The programming environment can be embedded in, or “plugged into”, an existing software development environment, or provided on its own. A syntax may be specified with source code statements, XML, SQL database definitions, a datastream, or any other derivative of the disclosed BNF grammar. The platform is an event based environment wherein events of modifying application data containing configured sought values/information are recognized for triggering processing of actions. Events can be defined with complex expressions, and actions can be defined using homegrown executables, APIs, scripts, applications, a set of commands provided with the LBX platform, or any other executable processing. The LBX platform includes a variety of embodiments as described.

Another advantage is providing a comprehensive palette of paste commands for pasting LBX data into data entry fields, snapshot images, or one or more video stream frames. Data can be accessed and used for pasting from: queue **22**; history **30**; statistics **14**; service directory **16**; atomic terms; map terms; WDRTerm data; AppTerm data; any term or construct of the LBX BNF grammar; data describing current, past or future LBX data; averages of MS or LBX data; data derived from MSs in the vicinity (e.g. nearby); and data sensed, received, sent, processed, analyzed, or predicted at the MS. Data being pasted may be converted prior to the paste as a user requests. The user may adjust the paste data appearance (font, size, color, or any other appearance characteristic) prior to finalizing the paste action.

Yet another advantage is providing “plug-in” application support so that an application can be integrated conveniently into the LBX architecture and framework through Prefix Registry Records **5300**. Application data and executable interfaces are “plugged in”. Application data is made accessible to charter processing for conditional and configurable event based charter processing. Various “plug-in” systems and methods are described. The LBX platform is designed to integrate well with MS applications of all varieties for a cohesive architecture.

Another advantage is for tightly coupling/integrating LBX processing configuration and processing into a programming environment for a WPL in context of a rich PPL. LBX processing can be a “plug-in” to PPLs, or may be integrated into the PPL syntax for a rich WPL. There are a variety of systems and methods described for a comprehensive LBX platform.

It is an advantage for facilitating the creation of charters that make sense in context of a particular MS application by automating suggestions. Special terms and atomic operands are determined for an application context, and candidate charters and/or portions thereof are presented for use to the user based on being derived from the special terms and atomic operands determined for the application context. A user’s effort in creating charters for a particular application context is minimized with ready-made charters or charter portions that are automatically determined to be relevant for the particular application context. Upon being presented with suggestions, the user can select, or select and “tweak” to a desired charter configuration. The user can also configure privileges that are in context of the application or the charters selected.

It is an advantage for automatically comparing MS data profile information for matches for triggering conditional actions of charters. Users can configure data which is beaconed to other MSs and then compared for matches for automated charter processing. MSs are automated with social interaction to other MSs so that MS users are alerted of MS users of interest in the vicinity for a variety of applications.

It is an advantage for transmitting application data fields to peer MSs in the vicinity, receiving application data fields from peer MSs in the vicinity, transmitting application data fields to data processing systems in the vicinity in a peer to peer manner, and receiving application data fields from data processing systems in the vicinity in a peer to peer manner for interoperability of a diverse set of applications and automated triggered processing thereof, while not using an application server to middle-man the data (e.g. MSs communicate with each other directly and wirelessly as peers). Application data fields shared between peer data processing systems (e.g. MSs) are preferably additionally available at a MS as AppTerm data (see below). A user has control for disabling or enabling which application data fields are shared. Privileges configured between MSs enforce desired effects for processing the data on MSs which send or receive the data.

A further advantage is to provide MSs with a wealth of location based enhanced applications without requiring a service. It is also an advantage to not require a service for geo-fence alerts, proactive content delivery, and nearby alerts, for example as described by server based U.S. patent pending Ser. No. 11/207,080 (“System and Method for Anonymous Location Based Services”, Johnson). Herein, alert processing, geo-fences and content is maintained at a MS for a) being processed at the MS when interacting directly with peer MSs; and b) being shared with peer MSs for being processed at peer MSs. Better performance of processing content delivery and providing alerts is achieved because it occurs at the MSs without any interoperability to some “middleman” service.

Another advantage is in leveraging the multi-threaded and wireless multi-wave, multi-frequency and multi-channel capability of the disclosed MS for RFID and RDS integration. RFID and RDS interfaces fit nicely in the LBX framework as described below.

A further advantage is for the MS to automatically, or upon user request, analyze a picture, or video stream frame, for the purpose of more confidently determining a MS location. User configurations are used to drive desired processing.

Another advantage is for thoroughly maintaining and managing statistics and history information at a MS. Many options are supported for how, where, and when to save such information.

A further advantage is to provide Sudden Proximal User Interfaces (SPUIs) at a MS when detecting other data processing systems in the vicinity (e.g. another MS, a RFID device, a data processing system emulating a MS, or any other data processing system). A SPUI is a GUI for notifying a MS user that a remote data processing system of interest is in the vicinity, based on configured “in the vicinity” conditions. Presenting the SPUI at the MS can be triggered by charter configurations, application term (AppTerm) trigger configurations, or RFID trigger configurations. There are many applications for SPUI processing for saving MS users time from MS user interface interactions for common tasks, for example appliance and device interfaces. Authentication can be automated. Also, SPUIs save data from previous executions for defaulting data in a subsequent execution thereby preventing the burdening of a MS user from re-entering data to the MS that was already entered once previously. There are many applications that fit within the SPUI framework, some of which are described below.

Another advantage is for providing a user with the ability to manually request to send/transmit outbound data with options for customizing, such as: a WDR, a derivative of a WDR, a subset of a WDR, a user configured set of data, or any customized set of data. If a WDR or derivative/subset thereof is to be sent, the WDR may first be searched for at the MS with user specified search criteria and/or transmitted outbound according to user specified transmission criteria.

It is an advantage to provide a task monitor/trace interface for examining MS task status for current and past system states. The task monitor interface permits convenient contextual charter creation as desired by the user based on task status findings.

It is an advantage for providing generic application record sorting based on: MS whereabouts, whereabouts of a particular MS, whereabouts of others in the vicinity, or other WDR search criteria for sorting WDRs maintained at the MS where the sort is requested.

Another advantage is for providing one or more vicinity monitors for indicating MSs of interest that are nearby. The multi-threaded MS supports a plurality of vicinity monitors. A MS user configures criteria/conditions (i.e. expression) for a vicinity monitor for being compared to WDR information as it is received at the MS. The expression result (True/False) determines whether or not the MS that originated the WDR is to be monitored within the particular vicinity monitor. A polling or asynchronous event (e.g. as WDRs received) design may be used.

Another advantage is for automatic inventory management processing for inventory items that are in the vicinity of a MS at some point in time. A MS user can move to the whereabouts of particular items he desires to keep an inventory of for automatically managing the inventory by counting the current stock, performing orders for stocking, and tracking an order. The MS user can configure payment information for automatic order processing. Inventory items are enabled for inventory management in having an associated data processing system (e.g. (RFID tag, affixed/integrated MS, etc). A MS user can manually perform an order using the automatically determined inventory count information, or the order can be scheduled for automatic ordering (e.g. using a calendar entry). Inventory items can be ordered individually or as a group, perhaps as part of a group hierarchy. Typical uses are for managing the life of a typical MS user: products stocked in kitchen pantry, refrigerator, freezer, closet, office, bathrooms, laundry room, office supply closet, or other areas of a MS user’s home, office or place of work.

Another advantage is for providing a MS user with a convenient resource mapping of privileges and charters between identities. For example, it could be tedious figuring out all the privileges, grants and charters which are granted to one MS user, and then granting those same rights to another MS user. Such a task is error prone and time consuming. Resource mapper functionality is provided wherein all rights (e.g. privileges) of one identifier can be assigned to another identifier in a single operation. The same rights can subsequently be removed as a single operation. A MS user has the ability to model granting privileges and charters to an identity (e.g. group), and then assign all of those, or remove all of those, in a single operation to other identifiers.

A further advantage is for different applications to be correlated through cross application addressing so that features or contexts of one application can be used to automatically affect features or contexts of another application. Identifiers used in context of one application are correlated to another application form. For example, an email application recipient address is correlated to the phone application caller id for the same MS in order to instantly (upon user request) show all emails associated to a person on an active phone call. The correlation occurs transparently without needing to know addressing details. There can be many identifier forms for correlation for a single MS depending on an application in use.

Further features and advantages of the disclosure, as well as the structure and operation of various embodiments of the disclosure, are described in detail below with reference to the accompanying drawings. In the drawings, like reference numbers generally indicate identical, functionally similar, and/or structurally similar elements. The drawing in which an element first appears is indicated by the leftmost digit(s) in the corresponding reference number, except that reference numbers 1 through 99 may be found on the first 4 drawings of FIGS. **1**A through **1**D, and FIG. **1**F. Dashed outlines (e.g. process blocks, data record fields) may be used in the drawings to highlight, or indicate optional embodiments, for example depending on MS performance considerations. None of the drawings, discussions, or materials herein is to be interpreted as limiting to a particular embodiment. The broadest interpretation is intended. Other embodiments accomplishing same functionality are within the spirit and scope of this disclosure. It should be understood that information is presented by example and many embodiments exist without departing from the spirit and scope of this disclosure.

BRIEF DESCRIPTION OF THE DRAWINGS

There is no guarantee that there are descriptions in this specification for explaining every novel feature found in the drawings. The present disclosure will be described with reference to the accompanying drawings, wherein:

FIG. **1**A depicts a preferred embodiment high level example componentization of a MS in accordance with the present disclosure;

FIG. **1**B depicts a Location Based eXchanges (LBX) architectural illustration for discussing the present disclosure;

FIG. **1**C depicts a Location Based Services (LBS) architectural illustration for discussing prior art of the present disclosure;

FIG. **1**D depicts a block diagram of a data processing system useful for implementing a MS, ILM, DLM, centralized server, or any other data processing system disclosed herein;

FIG. **1**E depicts a network illustration for discussing various deployments of whereabouts processing aspects of the present disclosure;

FIG. **1**F depicts a network illustration for discussing LBX character provided to a MS through user LBX configurations made;

FIG. **2**A depicts an illustration for describing automatic location of a MS through the MS coming into range of a stationary cellular tower;

FIG. **2**B depicts an illustration for describing automatic location of a MS through the MS coming into range of some stationary antenna;

FIG. **2**C depicts an illustration for discussing an example of automatically locating a MS through the MS coming into range of some stationary antenna;

FIG. **2**D depicts a flowchart for describing a preferred embodiment of a service whereabouts update event of an antenna in-range detected MS when MS location awareness is monitored by a stationary antenna or cell tower;

FIG. **2**E depicts a flowchart for describing a preferred embodiment of an MS whereabouts update event of an antenna in-range detected MS when MS location awareness is monitored by the MS;

FIG. **2**F depicts a flowchart for describing a preferred embodiment of a procedure for inserting a Whereabouts Data Record (WDR) to an MS whereabouts data queue;

FIG. **3**A depicts a locating by triangulation illustration for discussing automatic location of a MS;

FIG. **3**B depicts a flowchart for describing a preferred embodiment of the whereabouts update event of a triangulated MS when MS location awareness is monitored by some remote service;

FIG. **3**C depicts a flowchart for describing a preferred embodiment of the whereabouts update event of a triangulated MS when MS location awareness is monitored by the MS;

FIG. **4**A depicts a locating by GPS triangulation illustration for discussing automatic location of a MS;

FIG. **4**B depicts a flowchart for describing a preferred embodiment of the whereabouts update event of a GPS triangulated MS;

FIG. **5**A depicts a locating by stationary antenna triangulation illustration for discussing automatic location of a MS;

FIG. **5**B depicts a flowchart for describing a preferred embodiment of the whereabouts update event of a stationary antenna triangulated MS;

FIG. **6**A depicts a flowchart for describing a preferred embodiment of a service whereabouts update event of a physically or logically connected MS;

FIG. **6**B depicts a flowchart for describing a preferred embodiment of a MS whereabouts update event of a physically or logically connected MS;

FIGS. **7**A, **7**B and **7**C depict a locating by image sensory illustration for discussing automatic location of a MS;

FIG. **7**D depicts a flowchart for describing a preferred embodiment of graphically locating a MS, for example as illustrated by FIGS. **7**A through **7**C;

FIG. **8**A heterogeneously depicts a locating by arbitrary wave spectrum illustration for discussing automatic location of a MS;

FIG. **8**B depicts a flowchart for describing a preferred embodiment of locating a MS through physically contacting the MS;

FIG. **8**C depicts a flowchart for describing a preferred embodiment of locating a MS through a manually entered whereabouts of the MS;

FIG. **9**A depicts a table for illustrating heterogeneously locating a MS;

FIG. **9**B depicts a flowchart for describing a preferred embodiment of heterogeneously locating a MS;

FIGS. **10**A and **10**B depict an illustration of a Locatable Network expanse (LN-Expanse) for describing locating of an ILM with all DLMs;

FIG. **10**C depicts an illustration of a Locatable Network expanse (LN-Expanse) for describing locating of an ILM with an ILM and DLM;

FIGS. **10**D, **10**E, and **10**F depict an illustration of a Locatable Network expanse (LN-Expanse) for describing locating of an ILM with all ILMs;

FIGS. **10**G and **10**H depict an illustration for describing the infinite reach of a Locatable Network expanse (LN-Expanse) according to MSs;

FIG. **10**I depicts an illustration of a Locatable Network expanse (LN-Expanse) for describing a supervisory service;

FIG. **11**A depicts a preferred embodiment of a Whereabouts Data Record (WDR) **1100** for discussing operations of the present disclosure;

FIGS. **11**B, **11**C and **11**D depict an illustration for describing various embodiments for determining the whereabouts of an MS;

FIG. **11**E depicts an illustration for describing various embodiments for automatically determining the whereabouts of an MS;

FIG. **12** depicts a flowchart for describing an embodiment of MS initialization processing;

FIGS. **13**A through **13**C depict an illustration of data processing system wireless data transmissions over some wave spectrum;

FIG. **14**A depicts a flowchart for describing a preferred embodiment of MS LBX configuration processing;

FIG. **14**B depicts a continued portion flowchart of FIG. **14**A for describing a preferred to embodiment of MS LBX configuration processing;

FIG. **15**A depicts a flowchart for describing a preferred embodiment of DLM role configuration processing;

FIG. **15**B depicts a flowchart for describing a preferred embodiment of ILM role configuration processing;

FIG. **15**C depicts a flowchart for describing a preferred embodiment of a procedure for Manage List processing;

FIG. **16** depicts a flowchart for describing a preferred embodiment of NTP use configuration processing;

FIG. **17** depicts a flowchart for describing a preferred embodiment of WDR maintenance processing;

FIG. **18** depicts a flowchart for describing a preferred embodiment of a procedure for variable configuration processing;

FIG. **19** depicts an illustration for describing a preferred embodiment multithreaded architecture of peer interaction processing of a MS in accordance with the present disclosure;

FIG. **20** depicts a flowchart for describing a preferred embodiment of MS whereabouts broadcast processing;

FIG. **21** depicts a flowchart for describing a preferred embodiment of MS whereabouts collection processing;

FIG. **22** depicts a flowchart for describing a preferred embodiment of MS whereabouts supervisor processing;

FIG. **23** depicts a flowchart for describing a preferred embodiment of MS timing determination processing;

FIG. **24**A depicts an illustration for describing a preferred embodiment of a thread request queue record;

FIG. **24**B depicts an illustration for describing a preferred embodiment of a correlation response queue record;

FIG. **24**C depicts an illustration for describing a preferred embodiment of a WDR request record;

FIG. **25** depicts a flowchart for describing a preferred embodiment of MS WDR request processing;

FIG. **26**A depicts a flowchart for describing a preferred embodiment of MS whereabouts determination processing;

FIG. **26**B depicts a flowchart for describing a preferred embodiment of processing for determining a highest possible confidence whereabouts;

FIG. **27**A depicts a flowchart for describing a preferred embodiment of queue prune processing;

FIG. **27**B depicts a flowchart for describing a preferred embodiment of setting confidence default values based on user experience;

FIG. **28** depicts a flowchart for describing a preferred embodiment of MS termination processing;

FIG. **29**A depicts a flowchart for describing a preferred embodiment of a process for starting a specified number of threads in a specified thread pool;

FIG. **29**B depicts a flowchart for describing a preferred embodiment of a procedure for terminating the process started by FIG. **29**A;

FIGS. **30**A through **30**B depict a preferred embodiment BNF grammar for variables, variable instantiations and common grammar for BNF grammars of permissions, groups and charters;

FIG. **30**C depicts a preferred embodiment BNF grammar for permissions and groups;

FIGS. **30**D through **30**E depict a preferred embodiment BNF grammar for charters;

FIGS. **31**A through **31**E depict a preferred embodiment set of command and operand candidates for Action Data Records (ADRs) facilitating discussing associated parameters of the ADRs of the present disclosure;

FIG. **32**A depicts a preferred embodiment of a National Language Support (NLS) directive command cross reference;

FIG. **32**B depicts a preferred embodiment of a NLS directive operand cross reference;

FIG. **33**A depicts a preferred embodiment American National Standards Institute (ANSI) X.409 encoding of the BNF grammar of FIGS. **30**A through **30**B for variables, variable instantiations and common grammar for BNF grammars of permissions and charters;

FIG. **33**B depicts a preferred embodiment ANSI X.409 encoding of the BNF grammar of FIG. **30**C for permissions and groups;

FIGS. **33**C-**1** and **33**C-**2** (both hereinafter referred to as FIG. **33**C) depict a preferred embodiment ANSI X.409 encoding of the BNF grammar of FIGS. **30**D through **30**E for charters;

FIGS. **34**A through **34**G depict preferred embodiment C programming source code header file contents, derived from the grammar of FIGS. **30**A through **30**E;

FIG. **35**A depicts a preferred embodiment of a Granting Data Record (GDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. **30**A through **30**E;

FIG. **35**B depicts a preferred embodiment of a Grant Data Record (GRTDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. **30**A through **30**E;

FIG. **35**C depicts a preferred embodiment of a Generic Assignment Data Record (GADR) for discussing operations of the present disclosure, derived from the grammar of FIGS. **30**A through **30**E;

FIG. **35**D depicts a preferred embodiment of a Privilege Data Record (PDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. **30**A through **30**E;

FIG. **35**E depicts a preferred embodiment of a Group Data Record (GRPDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. **30**A through **30**E;

FIG. **36**A depicts a preferred embodiment of a Description Data Record (DDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. **30**A through **30**E;

FIG. **36**B depicts a preferred embodiment of a History Data Record (HDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. **30**A through **30**E;

FIG. **36**C depicts a preferred embodiment of a Time specification Data Record (TDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. **30**A through **30**E;

FIG. **36**D depicts a preferred embodiment of a Variable Data Record (VDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. **30**A through **30**E;

FIG. **37**A depicts a preferred embodiment of a Charter Data Record (CDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. **30**A through **30**E;

FIG. **37**B depicts a preferred embodiment of an Action Data Record (ADR) for discussing operations of the present disclosure, derived from the grammar of FIGS. **30**A through **30**E;

FIG. **37**C depicts a preferred embodiment of a Parameter Data Record (PARMDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. **30**A through **30**E;

FIG. **37**D depicts a preferred embodiment of Charters Starters schema for discussing operations of the present disclosure;

FIG. **38** depicts a flowchart for describing a preferred embodiment of MS permissions configuration processing;

FIGS. **39**A through **39**B depict flowcharts for describing a preferred embodiment of MS user interface processing for permissions configuration;

FIGS. **40**A through **40**B depict flowcharts for describing a preferred embodiment of MS user interface processing for grants configuration;

FIGS. **41**A through **41**B depict flowcharts for describing a preferred embodiment of MS user interface processing for groups configuration;

FIG. **42** depicts a flowchart for describing a preferred embodiment of a procedure for viewing MS configuration information of others;

FIG. **43** depicts a flowchart for describing a preferred embodiment of a procedure for configuring MS acceptance of data from other MSs;

FIG. **44**A depicts a flowchart for describing a preferred embodiment of a procedure for sending MS data to another MS;

FIG. **44**B depicts a flowchart for describing a preferred embodiment of receiving MS configuration data from another MS;

FIG. **45**A depicts a flowchart for describing a preferred embodiment of MS charters configuration processing;

FIG. **45**B depicts a flowchart for describing a preferred embodiment of MS charter enablement and disablement processing;

FIGS. **46**A through **46**B depict flowcharts for describing a preferred embodiment of MS user interface processing for charters configuration;

FIGS. **47**A through **47**B depict flowcharts for describing a preferred embodiment of MS user interface processing for actions configuration;

FIGS. **48**A through **48**B depict flowcharts for describing a preferred embodiment of MS user interface processing for parameter information configuration;

FIG. **49**A depicts an illustration for preferred permission data characteristics in the present disclosure LBX architecture;

FIG. **49**B depicts an illustration for preferred charter data characteristics in the present disclosure LBX architecture;

FIGS. **50**A through **50**C depict an illustration of data processing system wireless data transmissions over some wave spectrum;

FIG. **51**A depicts an example of a source code syntactical encoding embodiment of permissions, derived from the grammar of FIGS. **30**A through **30**E;

FIG. **51**B depicts an example of a source code syntactical encoding embodiment of charters, derived from the grammar of FIGS. **30**A through **30**E;

FIG. **52** depicts another preferred embodiment C programming source code header file contents, derived from the grammar of FIGS. **30**A through **30**E;

FIG. **53** depicts a preferred embodiment of a Prefix Registry Record (PRR) for discussing operations of the present disclosure;

FIG. **54** depicts an example of an XML syntactical encoding embodiment of permissions and charters, derived from the BNF grammar of FIGS. **30**A through **30**E;

FIG. **55**A depicts a flowchart for describing a preferred embodiment of MS user interface processing for Prefix Registry Record (PRR) configuration;

FIG. **55**B depicts a flowchart for describing a preferred embodiment of Application Term (AppTerm) data modification;

FIG. **56** depicts a flowchart for appropriately processing an encoding embodiment of the BNF grammar of FIGS. **30**A through **30**E, in context for a variety of parser processing embodiments;

FIG. **57** depicts a flowchart for describing a preferred embodiment of WDR In-process Triggering Smarts (WITS) processing;

FIG. **58** depicts an illustration for granted data characteristics in the present disclosure LBX architecture;

FIG. **59** depicts a flowchart for describing a preferred embodiment of a procedure for enabling LBX features and functionality in accordance with a certain type of permissions;

FIG. **60** depicts a flowchart for describing a preferred embodiment of a procedure for performing LBX actions in accordance with a certain type of permissions;

FIG. **61** depicts a flowchart for describing a preferred embodiment of performing processing in accordance with configured charters;

FIG. **62** depicts a flowchart for describing a preferred embodiment of a procedure for performing an action corresponding to a configured command;

FIG. **63**A depicts a flowchart for describing a preferred embodiment of a procedure for Send command action processing;

FIGS. **63**B-**1** through **63**B-**7** depicts a matrix describing how to process some varieties of the Send command;

FIG. **63**C depicts a flowchart for describing one embodiment of a procedure for Send command action processing, as derived from the processing of FIG. **63**A;

FIG. **64**A depicts a flowchart for describing a preferred embodiment of a procedure for Notify command action processing;

FIGS. **64**B-**1** through **64**B-**4** depicts a matrix describing how to process some varieties of the Notify command;

FIG. **64**C depicts a flowchart for describing one embodiment of a procedure for Notify command action processing, as derived from the processing of FIG. **64**A;

FIG. **65**A depicts a flowchart for describing a preferred embodiment of a procedure for Compose command action processing;

FIGS. **65**B-**1** through **65**B-**7** depicts a matrix describing how to process some varieties of the Compose command;

FIG. **65**C depicts a flowchart for describing one embodiment of a procedure for Compose command action processing, as derived from the processing of FIG. **65**A;

FIG. **66**A depicts a flowchart for describing a preferred embodiment of a procedure for Connect command action processing;

FIGS. **66**B-**1** through **66**B-**2** depicts a matrix describing how to process some varieties of the Connect command;

FIG. **66**C depicts a flowchart for describing one embodiment of a procedure for Connect command action processing, as derived from the processing of FIG. **66**A;

FIG. **67**A depicts a flowchart for describing a preferred embodiment of a procedure for Find command action processing;

FIGS. **67**B-**1** through **67**B-**13** depicts a matrix describing how to process some varieties of the Find command;

FIG. **67**C depicts a flowchart for describing one embodiment of a procedure for Find command action processing, as derived from the processing of FIG. **67**A;

FIG. **68**A depicts a flowchart for describing a preferred embodiment of a procedure for Invoke command action processing;

FIGS. **68**B-**1** through **68**B-**5** depicts a matrix describing how to process some varieties of the Invoke command;

FIG. **68**C depicts a flowchart for describing one embodiment of a procedure for Invoke command action processing, as derived from the processing of FIG. **68**A;

FIG. **69**A depicts a flowchart for describing a preferred embodiment of a procedure for Copy command action processing;

FIGS. **69**B-**1** through **69**B-**14** depicts a matrix describing how to process some varieties of the Copy command;

FIG. **69**C depicts a flowchart for describing one embodiment of a procedure for Copy command action processing, as derived from the processing of FIG. **69**A;

FIG. **70**A depicts a flowchart for describing a preferred embodiment of a procedure for Discard command action processing;

FIGS. **70**B-**1** through **70**B-**11** depicts a matrix describing how to process some varieties of the Discard command;

FIG. **70**C depicts a flowchart for describing one embodiment of a procedure for Discard command action processing, as derived from the processing of FIG. **70**A;

FIG. **71**A depicts a flowchart for describing a preferred embodiment of a procedure for Move command action processing;

FIGS. **71**B-**1** through **71**B-**14** depicts a matrix describing how to process some varieties of the Move command;

FIG. **71**C depicts a flowchart for describing one embodiment of a procedure for Move command action processing, as derived from the processing of FIG. **71**A;

FIG. **72**A depicts a flowchart for describing a preferred embodiment of a procedure for Store command action processing;

FIGS. **72**B-**1** through **72**B-**5** depicts a matrix describing how to process some varieties of the Store command;

FIG. **72**C depicts a flowchart for describing one embodiment of a procedure for Store command action processing, as derived from the processing of FIG. **72**A;

FIG. **73**A depicts a flowchart for describing a preferred embodiment of a procedure for Administration command action processing;

FIGS. **73**B-**1** through **73**B-**7** depicts a matrix describing how to process some varieties of the Administration command;

FIG. **73**C depicts a flowchart for describing one embodiment of a procedure for Administration command action processing, as derived from the processing of FIG. **73**A;

FIG. **74**A depicts a flowchart for describing a preferred embodiment of a procedure for Change command action processing;

FIG. **74**C depicts a flowchart for describing one embodiment of a procedure for Change command action processing, as derived from the processing of FIG. **74**A;

FIG. **75**A depicts a flowchart for describing a preferred embodiment of a procedure for sending data to a remote MS;

FIG. **75**B depicts a flowchart for describing a preferred embodiment of processing for receiving execution data from another MS;

FIG. **76**A depicts a flowchart for describing a preferred embodiment of processing a special term information paste action at a MS;

FIG. **76**B-**1** illustrates a preferred embodiment of Application term interface processing;

FIG. **76**B-**2** illustrates an embodiment of Application term interface processing for applications not using a standardized LBX coding practice;

FIG. **76**B-**3** illustrates a preferred embodiment of charter invocation interface processing;

FIG. **76**C illustrates a preferred embodiment of Application term shared memory records;

FIG. **76**D depicts a flowchart for describing a preferred embodiment of processing for contextual charter creation;

FIG. **77** depicts a flowchart for describing a preferred embodiment of configuring data to be maintained to WDR Application Fields;

FIG. **78** depicts a simplified example of an XML syntactical encoding embodiment of a profile for the profile section of WDR Application Fields;

FIG. **79**A illustrates a branch subset of a tree structure;

FIG. **79**B illustrates a binary tree equivalent to the tree structure of FIG. **79**A which is used to support XML tag tree traversal processing;

FIG. **79**C depicts a preferred embodiment C programming source code structure for encoding a node in an internalized XML tree;

FIG. **79**D depicts a flowchart for describing a preferred embodiment of a procedure for profile match operator evaluation;

FIG. **80**A depicts a LBX application fields implementation status table;

FIGS. **80**B-**1** through **80**B-**4** (referred generally as FIG. **80**B) depict some section descriptions of registered LBX application fields;

FIG. **80**C depicts a flowchart for describing a preferred embodiment of a procedure for application fields section initialization processing;

FIG. **80**D depicts a flowchart for describing a preferred embodiment of MS Radio Frequency Identification (RFID) probe processing;

FIG. **80**E depicts a flowchart for describing a preferred embodiment of processing for receiving data from an RFID device;

FIG. **81**A depicts a flowchart for describing a preferred embodiment of processing for configuring criteria used by a MS to graphically locate itself;

FIG. **81**B depicts a flowchart for describing a preferred embodiment of processing for a MS to graphically locate itself;

FIG. **82**A depicts a flowchart for describing a preferred embodiment of processing for maintaining LBX history;

FIG. **82**B depicts a flowchart for describing a procedure to maintain information to LBX history;

FIG. **83**A depicts a flowchart for describing a preferred embodiment of processing for configuring LBX statistics;

FIG. **83**B depicts a flowchart for describing a procedure to maintain information to LBX statistics;

FIG. **84**A depicts a flowchart for describing a preferred embodiment of processing for configuring service propagation;

FIG. **84**B depicts a flowchart for describing a procedure to process application fields according to how they are enabled or disabled;

FIG. **85**A depicts a preferred embodiment of a Service Directory Record (SDR) for discussing operations of the present disclosure;

FIG. **85**B depicts a flowchart for describing a preferred embodiment of a procedure for processing a request for a propagated service;

FIG. **85**C depicts a flowchart for describing an example embodiment of MS application processing relevant for interfacing to a propagated service;

FIG. **85**D depicts a flowchart for describing a preferred embodiment of processing at a MS when receiving a request for a propagated service from a remote MS;

FIG. **85**E depicts a flowchart for describing a preferred embodiment of processing for an executable that updates service directory information;

FIG. **86**A depicts a flowchart for describing a preferred embodiment of processing for configuring the service informant;

FIG. **86**B depicts a flowchart for describing a preferred embodiment procedure to provide service informant processing;

FIG. **86**C depicts a preferred embodiment of a Service Informant Record (SIR) for discussing operations of the present disclosure;

FIG. **87**A depicts a flowchart for describing a preferred embodiment of Sudden Proximal User Interface (SPUI) processing;

FIG. **87**B illustrates different embodiments for discussing various application data processing systems which can be automatically controlled by a MS according to the present disclosure;

FIG. **87**C depicts a flowchart for describing a remote data processing system application environment covering an infinite number of MS controllable applications;

FIG. **88**A depicts a flowchart for describing a preferred embodiment of manually transmitting WDR information;

FIG. **88**B depicts a flowchart for describing a preferred embodiment of MS task monitor processing;

FIG. **89**A depicts a flowchart for describing a preferred embodiment of updating a MS global variable for the last time a MS input peripheral was acted upon by a MS user;

FIG. **90**A depicts a flowchart for a preferred embodiment for processing the request to specify a map term;

FIG. **90**B depicts a preferred embodiment of a Map Term Data Record (MTDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. **30**A through **30**E;

FIGS. **91**A through **91**B depict preferred data schema embodiments of automated inventory management for discussing operations of the present disclosure;

FIG. **91**C depicts a flowchart for a preferred embodiment for inventory management processing;

FIG. **91**D depicts a flowchart for a preferred embodiment of automatically processing whereabouts of inventory items in the vicinity of a MS;

FIG. **92**A depicts a flowchart for a preferred embodiment for inventory group management processing;

FIG. **92**B depicts a flowchart for a preferred embodiment for automatic order processing of inventory items according to a schedule;

FIG. **93**A depicts a flowchart for a preferred embodiment for payment method management processing;

FIG. **93**B depicts a flowchart for a preferred embodiment for pending inventory order management processing;

FIG. **94**A depicts a flowchart for a preferred embodiment of a procedure for automatically ordering inventory;

FIG. **94**B depicts a flowchart for a preferred embodiment for order services management processing;

FIG. **95**A depicts a preferred embodiment of a resource mapper record for resource mapper processing of the present disclosure;

FIG. **95**B depicts a flowchart for a preferred embodiment for automatic resource mapper processing;

FIG. **96**A depicts a flowchart for a preferred embodiment for automatic application sort index processing;

FIG. **96**B illustrates an example application use of sort index processing;

FIG. **97**A depicts a flowchart for a preferred embodiment for vicinity monitor configuration processing;

FIG. **97**B depicts a preferred embodiment of a Vicinity Monitor Data Record (VMDR) for discussing operations of vicinity monitor processing; and

FIG. **97**C depicts a flowchart for a preferred embodiment for vicinity monitor processing.

DETAILED DESCRIPTION OF THE INVENTION

With reference now to detail of the drawings, the present disclosure is described. Obvious error handling is omitted from the flowcharts in order to focus on the key aspects of the present disclosure. Obvious error handling includes database I/O errors, field validation errors, errors as the result of database table/data constraints or unique keys, data access errors, communications interface errors or packet collision, hardware failures, checksum validations, bit error detections/corrections, and any other error handling as well known to those skilled in the relevant art in context of this disclosure. A semicolon may be used in flowchart blocks to represent, and separate, multiple blocks of processing within a single physical block. This allows simpler flowcharts with less blocks in the drawings by placing multiple blocks of processing description in a single physical block of the flowchart. Flowchart processing is intended to be interpreted in the broadest sense by example, and not for limiting methods of accomplishing the same functionality. Preferably, field validation in the flowcharts checks for SQL injection attacks, communications protocol sniff and hack attacks, preventing of spoofing MS addresses, syntactical appropriateness, and semantics errors where appropriate. Disclosed user interface processing and/or screenshots are also preferred embodiment examples that can be implemented in other ways without departing from the spirit and scope of this disclosure. Alternative user interfaces (since this disclosure is not to be limiting) will use similar mechanisms, but may use different mechanisms without departing from the spirit and scope of this disclosure.

Locational terms such as whereabouts, location, position, area, destination, perimeter, radius, geofence, situational location, or any other related two or three dimensional locational term used herein to described position(s) and/or locations and/or whereabouts is to be interpreted in the broadest sense. Location field **1100***c* may include an area (e.g. on earth), a point (e.g. on earth), or a three dimensional bounds in space. In another example, a radius may define a sphere in space, rather than a circle in a plane. In some embodiments, a planet field forms part of the location (e.g. Earth, Mars, etc as part of field **1100***c*) for which other location information (e.g. latitude and longitude on Mars also part of field **1100***c*) is relative. In some embodiments, elevations (or altitudes) from known locatable point(s), distances from origin(s) in the universe, etc. can denote where exactly is a point of three dimensional space, or three dimensional sphere, area, or solid, is located. That same point can provide a mathematical reference to other points of the solid area/region in space. Descriptions for angles, pitches, rotations, etc from some reference point(s) may be further provided. Three dimensional areas/regions include a conical shape, cubical shape, spherical shape, pyramidal shape, irregular shapes, or any other shape either manipulated with a three dimensional graphic interface, or with mathematical model descriptions. Areas/regions in space can be occupied by a MS, passed through (e.g. by a traveler) by a MS, or referenced through configuration by a MS. In a three dimensional embodiment, nearby/nearness is determined in terms of three dimensional information, for example, a spherical radius around one MS intersecting a spherical radius around another MS. In a two dimensional embodiment, nearby/nearness is determined in terms of two dimensional information, for example, a circular radius around one MS intersecting a circular radius around another MS. Points can be specified as a point in a x-y-z plane, a point in polar coordinates, or the like, perhaps the center of a planet (e.g. Earth) or the Sun, some origin in the Universe, or any other origin for distinctly locating three dimensional location(s), positions, or whereabouts in space. Elevation (e.g. for earth, or some other planet, etc) may be useful to the three dimensional point of origin, and/or for the three dimensional region in space. A region in space may also be specified with connecting x-y-z coordinates together to bound the three dimensional region in space. There are many methods for representing a location (field **1100***c*) without departing from the spirit and scope of this disclosure. MSs, for example as carried by users, can travel by airplane through three dimensional areas/regions in space, or travel under the sea through three dimensional regions in space.

Various embodiments of communications between MSs, or an MS and service(s), will share channels (e.g. frequencies) to communicate, depending on when in effect. Sharing a channel will involve carrying recognizable and processable signature to distinguish transmissions for carrying data. Other embodiments of communications between MSs, or an MS and service(s), will use distinct channels to communicate, depending on when in effect. The number of channels that can be concurrently listened on and/or concurrently transmitted on by a data processing system will affect which embodiments are preferred. The number of usable channels will also affect which embodiments are preferred. This disclosure avoids unnecessary detail in different communication channel embodiments so as to not obfuscate novel material. Independent of various channel embodiments within the scope and spirit of the present disclosure, MSs communicate with other MSs in a peer to peer manner, in some aspects like automated walkie-talkies.

Novel features disclosed herein need not be provided as all or none. Certain features may be isolated in some MS embodiments, or may appear as any subset of features and functionality in other embodiments.

Location Based eXchanges (LBX) Architecture

FIG. **1**A depicts a preferred embodiment high level example componentization of a MS in accordance with the present disclosure. A MS **2** includes processing behavior referred to as LBX Character **4** and Other Character **32**. LBX character **4** provides processing behavior causing MS **2** to take on the character of a Location Based Exchange (LBX) MS according to the present disclosure. Other Character **32** provides processing behavior causing MS to take on character of prior art MSs in context of the type of MS. Other character **32** includes at least other processing code **34**, other processing data **36**, and other resources **38**, all of which are well known to those skilled in the art for prior art MSs. Other character **32** provides a MS user with a limited set of configurability and functionality. In some embodiments, LBX character **4** components may, or may not, make use of other character **32** components **34**, **36**, and **38**. Other character **32** components may, or may not, make use of LBX character **4** components **6** through **30**.

LBX character **4** preferably includes at least Peer Interaction Processing (PIP) code **6**, Peer Interaction Processing (PIP) data **8**, self management processing code **18**, self management processing data **20**, WDR queue **22**, send queue **24**, receive queue **26**, service informant code **28**, and LBX history **30**. Peer interaction processing (PIP) code **6** comprises executable code in software, firmware, or hardware form for carrying out LBX processing logic of the present disclosure when interacting with another MS. Peer interaction processing (PIP) data **8** comprises data maintained in any sort of memory of MS **2**, for example hardware memory, flash memory, hard disk memory, a removable memory device, or any other memory means accessible to MS **2**. PIP data **8** contains intelligence data for driving LBX processing logic of the present disclosure when interacting with other MSs. Self management processing code **18** comprises executable code in software, firmware, or hardware form for carrying out the local user interface LBX processing logic of the present disclosure. Self management processing data **20** contains intelligence data for driving processing logic of the present disclosure as disclosed for locally maintained LBX features. WDR queue **22** contains Whereabouts Data Records (WDRs) **1100**, and is a First-In-First-Out (FIFO) queue when considering housekeeping for pruning the queue to a reasonable trailing history of inserted entries (i.e. remove stale entries). WDR queue **22** is preferably designed with the ability of queue entry retrieval processing similar to Standard Query Language (SQL) querying, wherein one or more entries can be retrieved by querying with a conditional match on any data field(s) of WDR **1100** and returning lists of entries in order by an ascending or descending key on one or any ascending/descending ordered list of key fields.

All disclosed queues (e.g. **22**, **24**, **26**, **1980**, **1990** (See FIG. **19**), or any other queue) are implemented with an appropriate thread-safe means of queue entry peeking (makes copy of sought queue entry without removing), discarding, retrieval, insertion, and queue entry field sorted search processing. Queues are understood to have an associated implicit semaphore to ensure appropriate synchronous access to queue data in a multi-threaded environment to prevent data corruption and misuse. Such queue interfaces are well known in popular operating systems. In MS operating system environments which do not have an implicit semaphore protected queue scheme, queue accesses in the present disclosure flowcharts are to be understood to have a previous request to a queue-assigned semaphore lock prior to queue access, and a following release of the semaphore lock after queue access. Operating systems without semaphore control may use methods to achieve similar thread-safe synchronization functionality. Queue functionality may be accomplished with lists, arrays, databases (e.g. SQL) and other methodologies without departing from the spirit and scope of queue descriptions herein.

Queue **22** alternate embodiments may maintain a plurality of WDR queues which segregate WDRs **1100** by field(s) values to facilitate timely processing. WDR queue **22** may be at least two (2) separate queues: one for maintaining the MS **2** whereabouts, and one for maintaining whereabouts of other MSs. WDR queue **22** may be a single instance WDR **1100** in some embodiments which always contains the most current MS **2** whereabouts for use by MS **2** applications (may use a sister queue **22** for maintaining WDRs from remote MSs). At least one entry is to be maintained to WDR queue **22** at all times for MS **2** whereabouts.

Send queue **24** (Transmit (Tx) queue) is used to send communications data, for example as intended for a peer MS within the vicinity (e.g. nearby as indicated by maximum range **1306**) of the MS **2**. Receive queue **26** (Receive (Rx) queue) is used to receive communications data, for example from peer MSs within the vicinity (e.g. nearby as indicated by maximum range **1306**) of the MS **2**. Queues **24** and **26** may also each comprise a plurality of queues for segregating data thereon to facilitate performance in interfacing to the queues, in particular when different queue entry types and/or sizes are placed on the queue. A queue interface for sending/receiving data to/from the MS is optimal in a multi-threaded implementation to isolate communications transport layers to processing behind the send/receive queue interfaces, but alternate embodiments may send/receive data directly from a processing thread disclosed herein. Queues **22**, **24**, and/or **26** may be embodied as a purely data form, or SQL database, maintained at MS **2** in persistent storage, memory, or any other storage means. In some embodiments, queues **24** and **26** are not necessary since other character **32** will already have accessible resources for carrying out some LBX character **4** processing.

Queue embodiments may contain fixed length records, varying length records, pointers to fixed length records, or pointers to varying length records. If pointers are used, it is assumed that pointers may be dynamically allocated for record storage on insertions and freed upon record use after discards or retrievals.

As well known to those skilled in the art, when a thread sends on a queue **24** in anticipation of a corresponding response, there is correlation data in the data sent which is sought in a response received by a thread at queue **26** so the sent data is correlated with the received data. In a preferred embodiment, correlation is built using a round-robin generated sequence number placed in data for sending along with a unique MS identifier (MS ID). If data is not already encrypted in communications, the correlation can be encrypted. While the unique MS identifier (MS ID) may help the MS identify which (e.g. wireless) data is destined for it, correlation helps identify which data at the MS caused the response. Upon receipt of data from a responder at queue **26**, correlation processing uses the returned correlation (e.g. field **1100***m*) to correlate the sent and received data. In preferred embodiments, the sequence number is incremented each time prior to use to ensure a unique number, otherwise it may be difficult to know which data received is a response to which data was sent, in particular when many data packets are sent within seconds. When the sequence number reaches a maximum value (e.g. 2**32−1), then it is round-robinned to 0 and is incremented from there all over again. This assures proper correlation of data between the MS and responders over time. There are other correlation schemes (e.g. signatures, random number generation, checksum counting, bit patterns, date/time stamp derivatives) to accomplish correlation functionality. If send and receive queues of Other Character **32** are used, then correlation can be used in a similar manner to correlate a response with a request (i.e. a send with a receipt).

There may be good reason to conceal the MS ID when transmitting it wirelessly. In this embodiment, the MS ID is a dependable and recognizable derivative (e.g. a pseudo MS ID) that can be detected in communications traffic by the MS having the pseudo MS ID, while concealing the true MS ID. This would conceal the true MS ID from would-be hackers sniffing wireless protocol. The derivative can always be reliably the same for simplicity of being recognized by the MS while being difficult to associate to a particular MS. Further still, a more protected MS ID (from would-be hackers that take time to deduce how an MS ID is scrambled) can itself be a dynamically changing correlation anticipated in forthcoming communications traffic, thereby concealing the real MS ID (e.g. phone number or serial number), in particular when anticipating traffic in a response, yet still useful for directing responses back to the originating MS (with the pseudo MS ID (e.g. correlation)). A MS would know which correlation is anticipated in a response by saving it to local storage for use until it becomes used (i.e. correlated in a matching response), or becomes stale. In another embodiment, a correlation response queue (like CR queue **1990**) can be deployed to correlate responses with requests that contain different correlations for pseudo MS IDs. In all embodiments, the MS ID (or pseudo MS ID) of the present disclosure should enable targeting communications traffic to the MS.

Service informant code **28** comprises executable code in software, firmware, or hardware form for carrying out of informing a supervisory service. The present disclosure does not require a connected web service, but there are features for keeping a service informed with activities of MS LBX. Service informant code **28** can communicate as requested any data **8**, **20**, **22**, **24**, **26**, **30**, **36**, **38**, or any other data processed at MS **2**.

LBX history **30** contains historical data useful in maintaining at MS **2**, and possibly useful for informing a supervisory service through service informant code **28**. LBX History **30** preferably has an associated thread of processing for keeping it pruned to the satisfaction of a user of MS **2** (e.g. prefers to keep last 15 days of specified history data, and 30 days of another specified history data, etc). With a suitable user interface to MS **2**, a user may browse, manage, alter, delete, or add to LBX History **30** as is relevant to processing described herein. Service informant code **28** may be used to cause sending of an outbound email, SMS message, outbound data packet, or any other outbound communication in accordance with LBX of the MS.

PIP data **8** preferably includes at least permissions **10**, charters **12**, statistics **14**, and a service directory **16**. Permissions **10** are configured to grant permissions to other MS users for interacting the way the user of MS **2** desires for them to interact. Therefore, permissions **10** contain permissions granted from the MS **2** user to other MS users. In another embodiment, permissions **10** additionally, or alternatively, contain permissions granted from other MS users to the MS **2** user. Permissions are maintained completely local to the MS **2**. Charters **12** provide LBX behavior conditional expressions for how MSs should interact with MS **2**. Charters **12** are configured by the MS **2** user for other MS users. In another embodiment, charters **12** additionally, or alternatively, are configured by other MS users for the MS **2** user. Some charters expressions depend on permissions **10**. Statistics **14** are maintained at MS **2** for reflecting peer (MS) to peer (MS) interactions of interest that occurred at MS **2**. In another embodiment, statistics **14** additionally, or alternatively, reflect peer (MS) to peer (MS) interactions that occurred at other MSs, preferably depending on permissions **10**. Service informant code **28** may, or may not, inform a service of statistics **14** maintained. Service directory **16** includes routing entries for how MS **2** will find a sought service, or how another MS can find a sought service through MS **2**.

In some embodiments, any code (e.g. **6**, **18**, **28**, **34**, **38**) can access, manage, use, alter, or discard any data (e.g. **8**, **20**, **22**, **24**, **26**, **30**, **36**, **38**) of any other component in MS **2**. Other embodiments may choose to keep processing of LBX character **4** and other character **32** disjoint from each other. Rectangular component boundaries are logical component representations and do not have to delineate who has access to what. MS (also MSs) references discussed herein in context for the new and useful features and functionality disclosed is understood to be an MS **2** (MSs **2**).

FIG. **1**B depicts a Location Based eXchanges (LBX) architectural illustration for discussing the present disclosure. LBX MSs are peers to each other for locational features and functionality. An MS **2** communicates with other MSs without requiring a service for interaction. For example, FIG. **1**B depicts a wireless network **40** of five (5) MSs. Each is able to directly communicate with others that are in the vicinity (e.g. nearby as indicated by maximum range **1306**). In a preferred embodiment, communications are limited reliability wireless broadcast datagrams having recognizable data packet identifiers. In another embodiment, wireless communications are reliable transport protocols carried out by the MSs, such as TCP/IP. In other embodiments, usual communications data associated with other character **32** include new data (e.g. Communications Key **1304**) in transmissions for being recognized by MSs within the vicinity. For example, as an MS conventionally communicates, LBX data is added to the protocol so that other MSs in the vicinity can detect, access, and use the data. The advantage to this is that as MSs use wireless communications to carry out conventional behavior, new LBX behavior is provided by simply incorporating additional information (e.g. Communications Key **1304**) to existing communications.

Regardless of the embodiment, an MS **2** can communicate with any of its peers in the vicinity using methods described below. Regardless of the embodiment, a communication path **42** between any two MSs is understood to be potentially bidirectional, but certainly at least unidirectional. The bidirectional path **42** may use one communications method for one direction and a completely different communications method for the other, but ultimately each can communicate to each other. When considering that a path **42** comprises two unidirectional communications paths, there are N*(N−1) unidirectional paths for N MSs in a network **40**. For example, 10 MSs results in 90 (i.e. 10*9) one way paths of communications between all 10 MSs for enabling them to talk to each other. Sharing of the same signaling channels is preferred to minimize the number of MS threads listening on distinct channels. Flowcharts are understood to process at incredibly high processing speeds, in particular for timely communications processing. While the MSs are communicating wirelessly to each other, path **42** embodiments may involve any number of intermediary systems or communications methods, for example as discussed below with FIG. **1**E.

FIG. **1**C depicts a Location Based Services (LBS) architectural illustration for discussing prior art of the present disclosure. In order for a MS to interact for LBS with another MS, there is service architecture **44** for accomplishing the interaction. For example, to detect that MS **1** is nearby MS N, the service is indispensably involved in maintaining data and carrying out processing. For example, to detect that MS **1** is arriving to, or departing from, a geofenced perimeter area configured by MS N, the service was indispensably involved in maintaining data and carrying out processing. For example, for MS N to locate MS **1** on a live map, the service was indispensably involved in maintaining data and carrying out processing. In another example, to grant and revoke permissions from MS **1** to MS N, the service was indispensably involved in maintaining data and carrying out processing. While it is advantageous to require a single bidirectional path **46** for each MS (i.e. two unidirectional communications paths; (2*N) unidirectional paths for N MSs), there are severe requirements for service(s) when there are lots of MSs (i.e. when N is large). Wireless MSs have advanced beyond cell phones, and are capable of housing significant parallel processing, processing speed, increased wireless transmission speeds and distances, increased memory, and richer features.

FIG. **1**D depicts a block diagram of a data processing system useful for implementing a MS, ILM, DLM, centralized server, or any other data processing system described herein. An MS **2** is a data processing system **50**. Data processing system **50** includes at least one processor **52** (e.g. Central Processing Unit (CPU)) coupled to a bus **54**. Bus **54** may include a switch, or may in fact be a switch **54** to provide dedicated connectivity between components of data processing system **50**. Bus (and/or switch) **54** is a preferred embodiment coupling interface between data processing system **50** components. The data processing system **50** also includes main memory **56**, for example, random access memory (RAM). Memory **56** may include multiple memory cards, types, interfaces, and/or technologies. The data processing system **50** may include secondary storage devices **58** such as persistent storage **60**, and/or removable storage device **62**, for example as a compact disk, floppy diskette, USB flash, or the like, also connected to bus (or switch) **54**. In some embodiments, persistent storage devices could be remote to the data processing system **50** and coupled through an appropriate communications interface. Persistent storage **60** may include flash memory, disk drive memory, magnetic, charged, or bubble storage, and/or multiple interfaces and/or technologies, perhaps in software interface form of variables, a database, shared memory, etc.

The data processing system **50** may also include a display device interface **64** for driving a connected display device (not shown). The data processing system **50** may further include one or more input peripheral interface(s) **66** to input devices such as a keyboard, keypad, Personal Digital Assistant (PDA) writing implements, touch interfaces, mouse, voice interface, or the like. User input ("user input", "user events" and "user actions" used interchangeably) to the data processing system are inputs accepted by the input peripheral interface(s) **66**. The data processing system **50** may still further include one or more output peripheral interface(s) **68** to output devices such as a printer, facsimile device, or the like. Output peripherals may also be available via an appropriate interface.

Data processing system **50** will include communications interface(s) **70** for communicating to another data processing system **72** via analog signal waves, digital signal waves, infrared proximity, copper wire, optical fiber, or other wave spectrums described herein. A MS may have multiple communications interfaces **70** (e.g. cellular connectivity, 802.x, etc). Other data processing system **72** may be an MS. Other data processing system **72** may be a service. Other data processing system **72** is a service data processing system when MS **50** communicates to other data processing system **72** by way of service informant code **28**. In any case, the MS and other data processing system are said to be interoperating when communicating.

Data processing system programs (also called control logic) may be completely inherent in the processor(s) **52** being a customized semiconductor, or may be stored in main memory **56** for execution by processor(s) **52** as the result of a read-only memory (ROM) load (not shown), or may be loaded from a secondary storage device into main memory **56** for execution by processor(s) **52**. Such programs, when executed, enable the data processing system **50** to perform features of the present disclosure as discussed herein. Accordingly, such data processing system programs represent controllers of the data processing system.

In some embodiments, the disclosure is directed to a control logic program product comprising at least one processor **52** having control logic (software, firmware, hardware microcode) stored therein. The control logic, when executed by processor(s) **52**, causes the processor(s) **52** to provide functions of the disclosure as described herein. In another embodiment, this disclosure is implemented primarily in hardware, for example, using a prefabricated component state machine (or multiple state machines) in a semiconductor element such as a processor **52**.

Those skilled in the art will appreciate various modifications to the data processing system **50** without departing from the spirit and scope of this disclosure. A data processing system, and more particularly a MS, preferably has capability for many threads of simultaneous processing which provide control logic and/or processing. These threads can be embodied as time sliced threads of processing on a single hardware processor, multiple processors, multicore processors, Digital Signal Processors (DSPs), or the like, or combinations thereof. Such multi-threaded processing can concurrently serve large numbers of concurrent MS tasks. Concurrent processing may be provided with distinct hardware processing and/or as appropriate software driven time-sliced thread processing. Those skilled in the art recognize that having multiple threads of execution on an MS is accomplished in many different ways without departing from the spirit and scope of this disclosure. This disclosure strives to deploy software to existing MS hardware configurations, but the disclosed software can be deployed as burned-in microcode to new hardware of MSs.

Data processing aspects of drawings/flowcharts are preferably multi-threaded so that many MSs and applicable data processing systems are interfaced with in a timely and optimal manner. Data processing system **50** may also include its own clock mechanism (not shown), if not an interface to an atomic clock or other clock mechanism, to ensure an appropriately accurate measurement of time in order to appropriately carry out processing described below. In some embodiments, Network Time Protocol (NTP) is used to keep a consistent universal time for MSs and other data processing systems in communications with MSs. This is most advantageous to prevent unnecessary round-tripping of data between data processing systems to determine timing (e.g. Time Difference of Arrival (TDOA)) measurements. A NTP synchronized date/time stamp maintained in communications is compared by a receiving data processing system for comparing with its own NTP date/time stamp to measure TOA (time of arrival (i.e. time taken to arrive)). Of course, in the absence of NTP used by the sender and receiver, TOA is also calculated in a bidirectional transmission using correlation. In this disclosure, TOA measurements from one location technology are used for triangulating with TOA measurements from another location technology, not just for determining "how close". Therefore, TDOA terminology is generally used herein to refer to the most basic TOA measurement of a wave spectrum signal being the difference between when it was sent and when it was received. TDOA is also used to describe using the difference of such measurements to locate (triangulate). NTP use among participating systems has the advantage of a single unidirectional broadcast data packet containing all a receiving system requires to measure TDOA, by knowing when the data was sent (date/time stamp in packet) and when the data was received (signal detected and processed by receiving system). A NTP clock source (e.g. atomic clock) used in a network is to be reasonably granular to carry out measurements, and ensures participating MSs are updated timely according to anticipated time drifts of their own clocks. MS clocks should maintain time as accurately as possible to minimize drift and minimize how often resynchronization with a NTP clock source is required. There are many well known methods for accomplishing NTP, some which require dedicated thread(s) for NTP processing, and some which use certain data transmitted to and from a source to keep time in synch.

Those skilled in the art recognize that NTP accuracy depends on participating MS clocks and processing timing, as well as time server source(s). Radio wave connected NTP time server(s) is typically accurate to as granular as 1 millisecond. Global Positioning System (GPS) time servers provide accuracy as granular as 50 microseconds. GPS timing receivers provide accuracy to around 100 nanoseconds, but this may be reduced by timing latencies in time server operating systems. With advancements in hardware, microcode, and software, obvious improvements are being made to NTP. In NTP use embodiments of this disclosure, an appropriate synchronization of time is used for functional interoperability between MSs and other data processing systems using NTP. NTP is not required in this disclosure, but it is an advantage when in use.

LBX Directly Located Mobile Data Processing Systems (DLMs)

FIG. **1**E depicts a network illustration for discussing various deployments of whereabouts processing aspects of the present disclosure. In some embodiments, a cellular network cluster **102** and cellular network cluster **104** are parts of a larger cellular network. Cellular network cluster **102** contains a controller **106** and a plurality of base stations, shown generally as base stations **108**. Each base station covers a single cell of the cellular network cluster, and each base station **108** communicates through a wireless connection with the controller **106** for call processing, as is well known in the art. Wireless devices communicate via the nearest base station (i.e. the cell the device currently resides in), for example base station **108***b*. Roaming functionality is provided when a wireless device roams from one cell to another so that a session is properly maintained with proper signal strength. Controller **106** acts like a telephony switch when a wireless device roams across cells, and it communicates with controller **110** via a wireless connection so that a wireless device can also roam to other clusters over a larger geographical area. Controller **110** may be connected to a controller **112** in a cellular cluster through a physical connection, for example, copper wire, optical fiber, or the like. This enables cellular clusters to be great distances from each other. Controller **112** may in fact be connected with a physical connection to its base stations, shown generally as base stations **114**. Base stations may communicate directly with the controller **112**, for example, base station **114***e*. Base stations may communicate indirectly to the controller **112**, for example base station **114***a* by way of base station **114***d*. It is well known in the art that many options exist for enabling interoperating communications between controllers and base stations for the purpose of managing a cellular network. A cellular network cluster **116** may be located in a different country. Base controller **118** may communicate with controller **110** through a Public Service Telephone Network (PSTN) by way of a telephony switch **120**, PSTN **122**, and telephony switch **124**, respectively. Telephony switch **120** and telephony switch **124** may be private or public. In one cellular network embodiment of the present disclosure, the services execute at controllers, for example controller **110**. In some embodiments, the MS includes processing that executes at a wireless device, for example mobile laptop computer **126**, wireless telephone **128**, a personal digital assistant (PDA) **130**, an iPhone **170**, or the like. As the MS moves about, positional attributes are monitored for determining location. The MS may be handheld, or installed in a moving vehicle. Locating a wireless device using wireless techniques such as Time Difference of Arrival (TDOA) and Angle Of Arrival (AOA) are well known in the art. The service may also execute on a server computer accessible to controllers, for example server computer **132**, provided an appropriate timely connection exists between cellular network controller(s) and the server computer **132**. Wireless devices (i.e. MSs) are preferably known by a unique identifier, for example a phone number, caller id, device identifier, or like appropriate unique handle.

In another embodiment of the present disclosure, GPS satellites such as satellite **134**, satellite **136**, and satellite **138** provide information, as is well known in the art, to GPS devices on earth for triangulation locating of the GPS device. In this embodiment, a MS has integrated GPS functionality so that the MS monitors its positions. The MS is preferably known by a unique identifier, for example a phone number, caller id, device identifier, or like appropriate unique handle (e.g. network address).

In yet another embodiment of the present disclosure, a physically connected device, for example, telephone **140**, computer **142**, PDA **144**, telephone **146**, and fax machine **148**, may be newly physically connected to a network. Each is a MS, although the mobility is limited. Physical connections include copper wire, optical fiber, USB, or any other physical connection, by any communications protocol thereon. Devices are preferably known by a unique identifier, for example a phone number, caller id, device identifier, physical or logical network address, or like appropriate unique handle. The MS is detected for being newly located when physically connected. A service can be communicated to upon detecting connectivity. The service may execute at an Automatic Response Unit (ARU) **150**, a telephony switch, for example telephony switch **120**, a web server **152** (for example, connected through a gateway **154**), or a like data processing system that communicates with the MS in any of a variety of ways as well known to those skilled the art. MS detection may be a result of the MS initiating a communication with the service directly or indirectly. Thus, a user may connect his laptop to a hotel network, initiate a communication with the service, and the service determines that the user is in a different location than the previous communication. A local area network (LAN) **156** may contain a variety of connected devices, each an MS that later becomes connected to a local area network **158** at a different location, such as a PDA **160**, a server computer **162**, a printer **164**, an internet protocol telephone **166**, a computer **168**, or the like. Hard copy presentation could be made to printer **164** and fax **148**.

Current technology enables devices to communicate with each other, and other systems, through a variety of heterogeneous system and communication methods. Current technology allows executable processing to run on diverse devices and systems. Current technology allows communications between the devices and/or systems over a plethora of methodologies at close or long distance. Many technologies also exist for automatic locating of devices. It is well known how to have an interoperating communications system that comprises a plurality of individual systems communicating with each other with one or more protocols. As is further known in the art of developing software, executable processing of the present disclosure may be developed to run on a particular target data processing system in a particular manner, or customized at install time to execute on a particular data processing system in a particular manner.

FIG. **1**F depicts a network illustration for discussing LBX character **4** provided to a MS through LBX configurations made, for example with permissions **10** and/or charters **12**. FIG. **1**F exemplifies FIG. **1**B in how user configurations provide wits and a unique personality to a MS. LBX character **4** wits (see WITS below) enable a vast and diverse set of processing behavior for location based processing, even for identically manufactured MSs having identically available applications for use. Every MS **2** can be very different and distinguished from other MSs **2** depending on permissions **10** and charters **12** which are configured for driving WITS processing. For example, a MS **2***p* with “Hog” LBX character contains user configurations for selfishly leveraging the LN-expanse for being located while never providing information for others to be located. Hog MS **2***p* contains configurations that are rich and deep in functionality for the user of MS **2***p*, but provide little functionality for other MS users. A MS **2***q* with “Monkey” LBX character contains configurations for “fun and games” which are suitable for interacting with other MSs for primarily entertainment and playful purposes. Monkey MS **2***q* contains configurations that provide enjoyment to the MS user and his peers. A MS **2***r* with “Dog” LBX character contains configurations for “being everyone’s best friend” whereby MS **2***r* maintains configurations for helping others in accordance with any requests made on behalf of peer MSs. For example, the user of MS **2***r* is willing to unquestionably create configurations to keep LBX peers happy and to facilitate locational applications at other MSs. A MS **2***s* with “Cow” LBX character contains configurations for “existing to contribute” to the LN-Expanse by maintaining configurations for facilitating the locating of other MSs, and to interact with other MSs for the purpose of supporting locational applications at other MSs without being solicited for support. A MS **2***t* with “Tiger” LBX character contains user configurations which are “strictly business” and suitable for interacting with other MSs for primarily locational business purposes. Tiger MS **2** contains configurations for allowing business associates to interact, for example for letting a boss and team member know whereabouts, or alerting business associates of being nearby, or for automatically performing charter actions for the purpose of improving business activities. The richness of locational features and functionality provided by the LBX architecture enables a MS user to configure an infinite set of LBX character **4** for characterizing a MS and how it interacts with other MSs. Users exploit their own creativity for how their MSs should behave and what personalities their MS should have. The user’s MS becomes a broader reaching, and more impacting, personification of a user’s moving presence.

In some embodiments, an administrator or authorized user (e.g. parent) configures the MS for intended LBX character and use by the main MS user (e.g. child). Credentials such as a password, access code, user identifier and password, etc, or other authorization scheme may be used when accessing a disclosed configuration interface to limit configurability to certain users, types of users, or users with certain privileges.

FIG. **2**A depicts an illustration for describing automatic location of a MS, for example a DLM **200**, through the MS coming into range of a stationary cellular tower. A DLM **200**, or any of a variety of MSs, travels within range of a cell tower, for example cell tower **108***b*. The known cell tower location is used to automatically detect the location of the DLM **200**. In fact, any DLM that travels within the cell served by cell tower **108***b* is identified as the location of cell tower **108***b*. The confidence of a location of a DLM **200** is low when the cell coverage of cell tower **108***b* is large. In contrast, the confidence of a location of a DLM **200** is higher when the cell coverage of cell tower **108***b* is smaller. However, depending on the applications locating DLMs using this method, the locating can be quite acceptable. Location confidence is improved with a TDOA measurement for the elapsed time of communication between DLM **200** and cell tower to determine how close the MS is to the cell tower. Cell tower **108***b* can process all locating by itself, or with interoperability to other services as connected to cell tower **108***b* in FIG. **1**E. Cell tower **108***b* can communicate the location of DLM **200** to a service, to the DLM **200**, to other MSs within its coverage area, any combination thereof, or to any connected data processing system, or MS, of FIG. **1**E.

FIG. **2**B depicts an illustration for describing automatic location of a MS, for example a DLM **200**, through the MS coming into range of some stationary antenna. DLM **200**, or any of a variety of MSs, travels within range of a stationary antenna **202** that may be mounted to a stationary object **204**. The known antenna location is used to automatically detect the location of the DLM **200**. In fact, any DLM that travels within the coverage area served by antenna **202** is identified as the location of antenna **202**. The confidence of a location of a DLM **200** is low when the antenna coverage area of antenna **202** is large. In contrast, the confidence of a location of a DLM **200** is higher when the antenna coverage area of antenna **202** is smaller. However, depending on the applications locating DLMs using this method, the locating can be quite acceptable. Location confidence is improved with a TDOA measurement for the elapsed time of communication between DLM **200** and a particular antenna to determine how close the MS is to the antenna. Antenna **202** can process all locating by itself (with connected data processing system (not shown) as well known to those skilled in the art), or with interoperability to other services as connected to antenna **202**, for example with connectivity described in FIG. **1**E. Antenna **202** can be used to communicate the location of DLM **200** to a service, to the DLM **200**, to other MSs within its coverage area, any combination thereof, or to any connected data processing system, or MS, of FIG. **1**E.

FIG. **2**C depicts an illustration for discussing an example of automatically locating a MS, for example a DLM **200**, through the MS coming into range of some stationary antenna. DLM **200**, or any of a variety of MSs, travels within range of a stationary antenna **212** that may be mounted to a stationary object, such as building **210**. The known antenna location is used to automatically detect the location of the DLM **200**. In fact, any DLM that travels within the coverage area served by antenna **212** is identified as the location of antenna **212**. The confidence of a location of a DLM **200** is low when the antenna coverage area of antenna **212** is large. In contrast, the confidence of a location of a DLM **200** is higher when the antenna coverage area of antenna **212** is smaller. However, depending on the applications locating DLMs using this method, the locating can be quite acceptable. Location confidence is improved with a TDOA measurement as described above. Antenna **212** can process all locating by itself (with connected data processing system (not shown) as well known to those skilled in the art), or with interoperability to other services as connected to antenna **212**, for example with connectivity described in FIG. **1**E. Antenna **212** can be used to communicate the location of DLM **200** to a service, to the DLM **200**, to other MSs within its coverage area, any combination thereof, or to any connected data processing system, or MS, of FIG. **1**E.

Once DLM **200** is within the building **210**, a strategically placed antenna **216** with a desired detection range within the building is used to detect the DLM **200** coming into its proximity. Wall breakout **214** is used to see the antenna **216** through the building **210**. The known antenna **216** location is used to automatically detect the location of the DLM **200**. In fact, any DLM that travels within the coverage area served by antenna **216** is identified as the location of antenna **216**. The confidence of a location of a DLM **200** is low when the antenna coverage area of antenna **216** is large. In contrast, the confidence of a location of a DLM **200** is higher when the antenna coverage area of antenna **216** is smaller. Travels of DLM **200** can be limited by objects, pathways, or other limiting circumstances of traffic, to provide a higher confidence of location of DLM **200** when located by antenna **216**, or when located by any locating antenna described herein which detects MSs coming within range of its location. Location confidence is improved with a TDOA measurement as described above. Antenna **216** can process all locating by itself (with connected data processing system (not shown) as well known to those skilled in the art), or with interoperability to other services as connected to antenna **216**, for example with connectivity described in FIG. **1**E. Antenna **216** can be used to communicate the location of DLM **200** to a service, to the DLM **200**, to other MSs within its coverage area, any combination thereof, or to any connected data processing system, or MS, of FIG. **1**E.

Other in-range detection antennas of a FIG. **2**C embodiment may be strategically placed to facilitate warehouse operations such as in Kubler et al.

FIG. **2**D depicts a flowchart for describing a preferred embodiment of a service whereabouts update event of an antenna in-range detected MS, for example a DLM **200**, when MS location awareness is monitored by a stationary antenna, or cell tower (i.e. the service thereof). FIGS. **2**A through **2**C location detection processing are well known in the art. FIG. **2**D describes relevant processing for informing MSs of their own whereabouts. Processing begins at block **230** when a MS signal deserving a response has been received and continues to block **232** where the antenna or cell tower service has authenticated the MS signal. A MS signal can be received for processing by blocks **230** through **242** as the result of a continuous, or pulsed, broadcast or beaconing by the MS (FIG. **13**A), perhaps as part of usual communication protocol in progress for the MS (FIG. **13**A usual data **1302** with embedded Communications Key (CK) **1304**), or an MS response to continuous, or pulsed, broadcast or beaconing via the service connected antenna (FIG. **13**C). MS and/or service transmission can be appropriately correlated for a response (as described above) which additionally facilitates embodiments using TDOA measurements (time of communications between the MS and antenna, or cell tower) to determine at least how close is the MS in range (or use in conjunction with other data to triangulate the MS location). The MS is preferably authenticated by a unique MS identifier such as a phone number, address, name, serial number, or any other unique handle to the MS. In this, and any other embodiments disclosed, an MS may be authenticated using a group identifier handle indicating membership to a supported/known group deserving further processing. Authentication will preferably consult a database for authenticating that the MS is known. Block **232** continues to block **234** where the signal received is immediately responded back to the MS, via the antenna, containing at least correlation along with whereabouts information for a Whereabouts Data Record (WDR) **1100** associated with the antenna (or cell tower). Thereafter, the MS receives the correlated response containing new data at block **236** and completes a local whereabouts data record **1100** (i.e. WDR **1100**) using data received along with other data determined by the MS.

In another embodiment, blocks **232** through **234** are not required. A service connected antenna (or cell tower) periodically broadcasts its whereabouts (WDR info (e.g. FIG. **13**C)) and MSs in the vicinity use that directly at block **236**. The MS can choose to use only the confidence and location provided, or may determine a TDOA measurement for determining how close it is. If the date/time stamp field **1100*b*** indicates NTP is in use by the service, and the MS is also using NTP, then a TDOA measurement can be determined using the one unidirectional broadcast via the antenna by using the date/time stamp field **1100*b*** received with when the WDR information was received by the MS (subtract time difference and use known wave spectrum for distance). If either the service or MS is not NTP enabled, then a bidirectional correlated data flow between the service and MS is used to assess a TDOA measurement in terms of time of the MS. One embodiment provides the TDOA measurement from the service to the MS. Another embodiment calculates the TDOA measurement at the MS.

Network Time protocol (NTP) can ensure MSs have the same atomic clock time as the data processing systems driving antennas (or cell towers) they will encounter. Then, date/time stamps can be used in a single direction (unidirectional) broadcast packet to determine how long it took to arrive to/from the MS. In an NTP embodiment, the MS (FIG. **13**A) and/or the antenna (FIG. **13**C) sends a date/time stamp in the pulse, beacon, or protocol. Upon receipt, the antenna (or cell tower) service data processing system communicates how long the packet took from an MS to the antenna (or cell tower) by comparing the date/time stamp in the packet and a date/time stamp of when it was received. The service may also set the confidence value, before sending WDR information to the MS. Similarly, an MS can compare a date/time stamp in the unidirectional broadcast packet sent from a locating service (FIG. **13**C) with when received by the MS. So, NTP facilitates TDOA measurements in a single broadcast communication between systems through incorporation to usual communications data **1302** with a date/time stamp in Communications Key (CK) **1304**, or alternatively in new data **1302**. Similarly, NTP facilitates TDOA measurement in a single broadcast communication between systems through incorporation to usual communications data **1312** with a date/time stamp in Communications Key (CK) **1314**, or alternatively in new data **1312**.

The following template is used in this disclosure to highlight field settings. See FIG. **11**A descriptions. Fields are set to the following upon exit from block **236**:

MS ID field **1100***a* is preferably set with: Unique MS identifier of the MS invoking block **240**. This field is used to uniquely distinguish this MS WDRs on queue **22** from other originated WDRs.

DATE/TIME STAMP field **1100***b* is preferably set with: Date/time stamp for WDR completion at block **236** to the finest granulation of time achievable by the MS. The NTP use indicator is set appropriately.

LOCATION field **1100***c* is preferably set with: Location of stationary antenna (or cell tower) as communicated by the service to the MS.

CONFIDENCE field **1100***d* is preferably set with: The same value (e.g. 76) for any range within the antenna (or cell tower), or may be adjusted using the TDOA measurement (e.g. amount of time detected by the MS for the response at block **234**). The longer time it takes between the MS sending a signal detected at block **232** and the response with data back received by the MS (block **234**), the less confidence there is for being located because the MS must be a larger distance from the antenna or cell tower. The less time it takes between the MS sending a signal detected at block **232** and the response with data back, the more confidence there is for being located because the MS must be a closer distance to the antenna or cell tower. Confidence values are standardized for all location technologies. In some embodiments of FIG. **2**D processing, a confidence value can be set for 1 through 100 (1 being lowest confidence and 100 being highest confidence) wherein a unit of measurement between the MS and antenna (or cell tower) is used directly for the confidence value. For example, 20 meters is used as the unit of measurement. For each unit of 20 meters distance determined by the TDOA measurement, assign a value of 1, up to a worst case of 100 (i.e. 2000 meters). Round the 20 meter unit of distance such that 0 meters to <25 meters is 20 meters (i.e. 1 unit of measurement), 26 meters to <45 meters is 40 meters (i.e. 2 units of measurement), and so on. Once the number of units is determined, subtract that number from **101** for the confidence value (i.e. 1 unit=confidence value 100, 20 units=confidence value 81; 100 units or greater=confidence value of 1). Yet another embodiment will use a standard confidence value for this "coming in range" technology such as 76 and then further increase or decrease the confidence using the TDOA measurement. Many embodiments exist for quantifying a higher versus lower confidence. In any case, a confidence value (e.g. 76) is determined by the MS, service, or both (e.g. MS uses TDOA measurement to modify confidence sent by service).

LOCATION TECHNOLOGY field **1100***e* is preferably set with: "Server Antenna Range" for an antenna detecting the MS, and is set to "Server Cell Range" for a cell tower detecting the MS. The originator indicator is set to DLM.

LOCATION REFERENCE INFO field **1100***f* is preferably set with: The period of time for communications between the antenna and the MS (a TDOA measurement), if known; a communications signal strength, if available; wave spectrum used (e.g. from MS receive processing), if available; particular communications interface **70**, if available. The TDOA measurement may be converted to a distance using wave spectrum information. The values populated here should have already been factored into the confidence value at block **236**.

COMMUNICATIONS REFERENCE INFO field **1100***g* is preferably set with: Parameters uniquely identifying a/the service (e.g. antenna (or cell tower)) and how to best communicate with it again, if available. May not be set, regardless if received from the service.

SPEED field **1100***h* is preferably set with: Data received by MS at block **234**, if available.

HEADING field **1100***i* is preferably set with: Data received by MS at block **234**, if available.

ELEVATION field **1100***j* is preferably set with: data received by MS at block **234**, if available. Elevation field **1100***j* is preferably associated with the antenna (or cell tower) by the elevation/altitude of the antenna (or cell tower).

APPLICATION FIELDS field **1100***k* is preferably set with: Data received at block **234** by the MS, or set by data available to the MS, or set by both the locating service for the antenna (or cell tower) and the MS itself. Application fields include, and are not limited to, MS navigation APIs in use, social web site identifying information, application information for applications used, accessed, or in use by the MS, or any other information complementing whereabouts of the MS.

CORRELATION FIELD **1100***m* is preferably set with: Not Applicable (i.e. not maintained to queue **22**).

SENT DATE/TIME STAMP field **1100***n* is preferably set with: Not Applicable (i.e. not maintained to queue **22**).

RECEIVED DATE/TIME STAMP field **1100***p* is preferably set with: Not Applicable (i.e. not maintained to queue **22**).

A service connected to the antenna (or cell tower) preferably uses historical information and artificial intelligence interrogation of MS travels to determine fields **1100***h* and **1100***i*. Block **236** continues to block **238** where parameters are prepared for passing to FIG. **2**F processing invoked at block **240**. Parameters are set for: WDRREF=a reference or pointer to the WDR; DELETEQ=FIG. **2**D location queue discard processing; and SUPER=FIG. **2**D supervisory notification processing. Thereafter, block **240** invokes FIG. **2**F processing and FIG. **2**D processing terminates at block **242**. FIG. **2**F processing will insert to queue **22** so this MS knows at least its own whereabouts whenever possible. A single data instance embodiment of WDR queue **22** will cause FIG. **2**F to update the single record of WDR information for being current upon exit from block **240** (this is true for all flowchart blocks invoking FIG. **2**F processing).

With reference now to FIG. **2**F, depicted is a flowchart for describing a preferred embodiment of a procedure for inserting a Whereabouts Data Record (WDR) **1100** to MS WDR queue **22**. Appropriate semaphores are used for variables which can be accessed simultaneously by another thread other than the caller. With reference now to FIG. **2**F, procedure processing starts at block **270** and continues to block **272** where parameters passed from the invoking block of processing, for example block **240**, are determined. The variable WDRREF is set by the caller to a reference or pointer to the WDR so subsequent blocks of FIG. **2**F can access the WDR. The variable DELETEQ is set by the caller so that block **292** knows how to discard obsolete location queue entries. The DELETEQ variable can be a multi-field record (or reference thereof) for how to prune. The variable SUPER is set by the caller so that block **294** knows under what condition(s), and which data, to contact a supervisory service. The SUPER variable can be a multi-field record (or reference thereof) for instruction.

Block **272** continues to block **274** where the DLMV (see FIG. **12** and later discussions for DLMV (DLM role(s) List Variable)), or ILMV (see FIG. **12** and later discussions for ILMV (ILM role(s) List Variable)), is checked for an enabled role matching the WDR for insertion (e.g. DLM: location technology field **1100***e* (technology and originator indicator) when MS ID=this MS; ILM: DLM or ILM indicator when MS ID not this MS). If no corresponding DLMV/ILMV role is enabled for the WDR to insert, then processing continues to block **294** (the WDR is not inserted to queue **22**). If the ILMV/DLMV role for the WDR is enabled, then processing continues to block **276** where the confidence of the WDR **1100** is validated prior to insertion. An alternate embodiment to FIG. **2**F will not have block **274** (i.e. block **272** continues directly to block **276**) since appropriate DLM and/or ILM processing may be terminated anyway when DLM/ILM role(s) are disabled (see FIG. **14**A/B).

If block **276** determines the data to be inserted is not of acceptable confidence (e.g. field **1100***d*<confidence floor value (see FIG. **14**A/B)), then processing continues to block **294** described below. If block **276** determines the data to be inserted is of acceptable confidence (e.g. field **1100***d*>70), then processing continues to block **278** for checking the intent of the WDR insertion.

If block **278** determines the WDR for insert is a WDR describing whereabouts for this MS (i.e. MS ID matching MS of FIG. **2**F processing (DLM: FIGS. **2**A through **9**B, or ILM: FIG. **26**A/B)), then processing continues to block **280**. If block **278** determines the WDR for insert is from a remote ILM or DLM (i.e. MS ID does not match MS of FIG. **2**F processing), then processing continues to block **290**. Block **280** peeks the WDR queue **22** for the most recent highest confidence entry for this MS whereabouts by searching queue **22** for: the MS ID field **1100***a* matching the MS ID of FIG. **2**F processing, and a confidence field **1100***d* greater than or equal to the confidence floor value, and a most recent date/time stamp field **1100***b*. Thereafter, if block **282** determines one was found, then processing continues to block **284**, otherwise processing continues to block **286** where a Last Whereabouts date/Time stamp (LWT) variable is set to field **1100***b* of the WDR for insert (e.g. first MS whereabouts WDR), and processing continues to block **288**.

If block **284** determines the WDR for insertion has significantly moved (i.e. using a movement tolerance configuration (e.g. 3 meters) with fields **1100***c* of the WDR for insert and the WDR peeked at block **280**), then block **286** sets the LWT (Last Whereabouts date/Time stamp) variable (with appropriate semaphore) to field **1100***b* of the WDR for insert, and processing continues to block **288**, otherwise processing continues directly to block **288** (thereby keeping the LWT as its last setting). The LWT is to hold the most recent date/time stamp of when the MS significantly moved as defined by a movement tolerance. The movement tolerance can be system defined or configured, or user configured in FIG. **14** by an option for configuration detected at block **1408**, and then using the Configure Value procedure of FIG. **18** (like confidence floor value configuration).

Block **288** accesses the DLMV and updates it with a new DLM role if there is not one present for it. This ensures a correct list of DLMV roles are available for configuration by FIG. **14**. Preferably, by default an unanticipated DLMV role is enabled (helps inform the user of its availability). Likewise in another embodiment, ILMV roles can be similarly updated, in particular if a more granulated list embodiment is maintained to the ILMV, or if unanticipated results help to identify another configurable role. By default, block **274** should allow unanticipated roles to continue with WDR insertion processing, and then block **288** can add the role, enable it, and a user can decide what to do with it in configuration (FIG. **14**A/B).

Thereafter, the WDR **1100** is inserted to the WDR queue **22** at block **290**, block **292** discards any obsolete records from the queue as directed by the caller (invoker), and processing continues to block **294**. The WDR queue **22** preferably contains a list of historically MS maintained Whereabouts Data Records (WDRs) as the MS travels. When the MS needs its own location, for example from an application access, or to help locate an ILM, the queue is accessed for returning the WDR with the highest confidence value (field **1100***d*) in the most recent time (field **1100***b*) for the MS (field **1100***a*). Block **292** preferably discards by using fields **1100***b* and **1100***d* relative to other WDRs. The queue should not be allowed to get too large. This will affect memory (or storage) utilization at the MS as well as timeliness in accessing a sought queue entry. Block **292** also preferably discards WDRs from queue **22** by moving selected WDRs to LBX History **30**.

As described above, queue interfaces assume an implicit semaphore for properly accessing queue **22**. There may be ILMs requesting to be located, or local applications of the MS may request to access the MS whereabouts. Executable thread(s) at the MS can accesses the queue in a thread-safe manner for responding to those requests. The MS may also have multiple threads of processing for managing whereabouts information from DLMs, ILMs, or stationary location services. The more concurrently executable threads available to the MS, the better the MS is able to locate itself and respond to others (e.g. MSs). There can be many location systems and methods used to keeping a MS informed of its own whereabouts during travel. While the preferred embodiment is to maximize thread availability, the obvious minimum requirement is to have at least 1 executable thread available to the MS. As described above, in operating system environments without proper queue interfaces, queue access blocks are first preceded by an explicit request for a semaphore lock to access queue **22** (waits until obtained), and then followed by a block for releasing the semaphore lock to another thread for use. Also, in the present disclosure it is assumed in blocks which access data accessible to more than 1 concurrent thread (e.g. shared memory access to DLMV or ILMV at block **274**) that an appropriate semaphore (created at block **1220**) protect synchronous access.

If block **294** determines information (e.g. whereabouts) should be communicated by service informant code **28** to a supervisory service, for example a service **1050**, then block **296** communicates specified data to the service and processing terminates at block **298** by returning to the invoker (caller). If block **294** determines a supervisory service is not to be informed, then processing terminates with an appropriate return to the caller at block **298**. Service informant code **28**, at block **296**, can send information as data that is reliably acknowledged on receipt, or as a datagram which most likely (but unreliably) is received.

Depending on the SUPER variable, block **294** may opt to communicate every time a WDR is placed to the queue, or when a reasonable amount of time has passed since last communicating to the supervisory service, or when a WDR confidence reaches a certain sought value, or when any WDR field or fields contain certain sought information, or when a reasonably large number of entries exist in WDR queue **22**, or for any processing condition encountered by blocks **270** through **298**, or for any processing condition encountered by caller processing up to the invocation of FIG. **2**F processing. Different embodiments will send a single WDR **1100** at block **296**, a plurality of WDRs **1100**, or any other data. Various SUPER parameter(s) embodiments for FIG. **2**F caller parameters can indicate what, when, where and how to send certain data. Block **296** may send an email, an SMS message, or use other means for conveying data. Service informant code **28** may send LBX history **30**, statistics **14** and/or any other data **8**, data **20**, queue data, data **36** or resources **38**. Service informant code **28** may update data in history **30**, statistics **14** or any other data **8**, data **20**, queue data, data **36** and/or resources **38**, possibly using conditions of this data to determine what is updated. Blocks **294** and **296** may be omitted in some embodiments.

If a single WDR is sent at block **296** as passed to FIG. **2**F processing, then the WDR parameter determined at block **272** is accessed. If a plurality of WDRs is sent at block **296**, then block **296** appropriately interfaces in a thread-safe manner to queue **22**, and sends the WDRs.

Some preferred embodiments do not incorporate blocks **278** through **286**. (i.e. block **276** continues to block **288** if confidence ok). Blocks **278** through **286** are for the purpose of implementing maintaining a date/time stamp of last MS significant movement (using a movement tolerance). Architecture **1900** uses FIG. **2**F, as does DLM processing. FIG. **2**F must perform well for the preferred multithreaded architecture **1900**. Block **280** performs a peek, and block **284** can be quite timely depending on embodiments used for location field **1100***c*. A movement tolerance incorporated at the MS is not necessary, but may be nice to have. Therefore, blocks **278** through **286** are optional blocks of processing.

FIG. **2**F may also maintain (with appropriate semaphore) the most recent WDR describing whereabouts of the MS of FIG. **2**F processing to a single data record every time a new one is to be inserted. This allows applications needing current whereabouts to simply access a current WDR, rather than interface to a plurality of WDRs at queue **22**. For example, there could be a new block **289** for updating the single WDR **1100** (just prior to block **290** such that incoming blocks to block **290** go to new block **289**, and new block **289** continues to block **290**).

With reference now to FIG. **2**E, depicted is a flowchart for describing a preferred embodiment of an MS whereabouts update event of an antenna in-range detected MS, for example a DLM **200**, when MS location awareness is monitored by the MS. FIG. **2**E describes relevant processing for MSs to maintain their own whereabouts. Processing begins at block **250** when the MS receives a signal from an antenna (or cell tower) deserving a response and continues to block **252** where the antenna or cell tower signal is authenticated by the MS as being a legitimate signal for processing. The signal can be received for processing by blocks **250** through **264** as the result of a continuous, or pulsed, broadcast or beaconing by the antenna, or cell tower (FIG. **13**C), or as part of usual communication protocol in progress with at least one MS (FIG. **13**C usual data **1312** with embedded Communications Key **1314**), or as a response via antenna to a previous MS signal (FIG. **13**A). The signal is preferably authenticated by a data parsed signature deserving further processing. Block **252** continues to block **254** where the MS sends an outbound request for soliciting an immediate response from the antenna (or cell tower) service. The request by the MS is appropriately correlated (e.g. as described above) for a response, which additionally facilitates embodiments using TDOA measurements (time of communications between the MS and antenna, or cell tower) to determine how close is the MS in range. Block **254** waits for a response, or waits until a reasonable timeout, whichever occurs first. There are also multithreaded embodiments to breaking up FIG. **2**E where block **254** does not wait, but rather terminates FIG. **2**E processing and depends on another thread to correlate the response and then continue processing blocks **256** through **260** (like architecture **1900**).

Thereafter, if block **256** determines the request timed out, then processing terminates at block **264**. If block **256** determines the response was received, then processing continues to block **258**. Block **258** completes a WDR **1100** with appropriate response data received along with data set by the MS. See FIG. **11**A descriptions. Fields are set to the following upon exit from block **258**:

MS ID field **1100***a* is preferably set with: Same as was described for FIG. **2**D (block **236**) above.

DATE/TIME STAMP field **1100***b* is preferably set with: Same as was described for FIG. **2**D (block **236**) above.

LOCATION field **1100***c* is preferably set with: Same as was described for FIG. **2**D (block **236**) above.

CONFIDENCE field **1100***d* is preferably set with: Same as was described for FIG. **2**D (block **236**) above.

LOCATION TECHNOLOGY field **1100***e* is preferably set with: "Client Antenna Range" for an antenna detecting the MS, and is set to "Client Cell Range" for a cell tower detecting the MS. The originator indicator is set to DLM.

LOCATION REFERENCE INFO field **1100***f* is preferably set with: Same as was described for FIG. **2**D (block **236**) above.

COMMUNICATIONS REFERENCE INFO field **1100***g* is preferably set with: Same as was described for FIG. **2**D (block **236**) above.

SPEED field **1100***h* is preferably set with: Same as was described for FIG. **2**D (block **236**) above.

HEADING field **1100***i* is preferably set with: Same as was described for FIG. **2**D (block **236**) above.

ELEVATION field **1100***j* is preferably set with: Same as was described for FIG. **2**D (block **236**) above.

APPLICATION FIELDS field **1100***k* is preferably set with: Same as was described for FIG. **2**D (block **236**) above.

CORRELATION FIELD **1100***m* is preferably set with: Not Applicable (i.e. not maintained to queue **22**).

SENT DATE/TIME STAMP field **1100***n* is preferably set with: Not Applicable (i.e. not maintained to queue **22**).

RECEIVED DATE/TIME STAMP field **1100***p* is preferably set with: Not Applicable (i.e. not maintained to queue **22**).

The longer time it takes between sending a request and getting a response at block **254**, the less confidence there is for being located because the MS must be a larger distance from the antenna or cell tower. The less time it takes, the more confidence there is for being located because the MS must be a closer distance to the antenna or cell tower.

Confidence values are analogously determined as described for FIG. **2**D. FIG. **2**D NTP embodiments also apply here. NTP can be used so no bidirectional communications is required for TDOA measurement. In this embodiment, the antenna (or cell tower) sets a NTP date/time stamp in the pulse, beacon, or protocol. Upon receipt, the MS instantly knows how long the packet took to be received by comparing the NTP date/time stamp in the packet and a MS NTP date/time stamp of when it was received (i.e. no request/response pair required). If location information is also present with the NTP date/time stamp in data received at block **252**, then block **252** can continue directly to block **258**.

An alternate MS embodiment determines its own (direction) heading and/or speed for WDR completion based on historical records maintained to the WDR queue **22** and/or LBX history **30**.

Block **258** continues to block **260** for preparing parameters for: WDRREF=a reference or pointer to the WDR; DELETEQ=FIG. **2**E location queue discard processing; and SUPER=FIG. **2**E supervisory notification processing. Thereafter, block **262** invokes the procedure (FIG. **2**F processing) to insert the WDR to queue **22**. After FIG. **2**F processing of block **262**, FIG. **2**E processing terminates at block **264**.

In alternative “coming within range” (same as “in range”, “in-range”, “within range”) embodiments, a unique MS identifier, or MS group identifier, for authenticating an MS for locating the MS is not necessary. An antenna emitting signals (FIG. **13**C) will broadcast (in CK **1314** of data **1312**) not only its own location information (e.g. location field **1100***c*), but also an NTP indicated date/time stamp field **1100***b*, which the receiving MS (also having NTP for time synchronization) uses to perform a TDOA measurement upon receipt. This will enable a MS to determine at least how close (e.g. radius **1318** range, radius **1320** range, radius **1322** range, or radius **1316** range) it is located to the location of the antenna by listening for and receiving the broadcast (e.g. of FIG. **13**C). Similarly, in another embodiment, an NTP synchronized MS emits signals (FIG. **13**A) and an NTP synchronized data processing system associated with a receiving antenna can make a TDOA measurement upon signal receipt. In other embodiments, more than a single unidirectional signal may be used while still preventing the requirement to recognize the MS to locate it. For example, an antenna emitting signals (e.g. FIG. **13**C hotspot WiFi 802.x) will contain enough information for a MS to respond with correlation for being located, and visa-versa. In any case, there can be multi-directional exchanged signals for determining a TDOA measurement.

FIG. **3**A depicts a locating by triangulation illustration for discussing automatic location of a MS, for example DLM **200**. DLM **200** is located through triangulation, as is well known in the art. At least three base towers, for example, base tower **108***b*, base tower **108***d*, and base tower **108***f*, are used for locating the MS. A fourth base tower may be used if elevation (or altitude) was configured for use in locating DLM **200**. There are cases where only two base towers are necessary given routes of travel are limited and known, for example, in spread out roadways or limited configured locations. Base towers may also be antennas **108***b*, **108***d*, and **108***f* in similar triangulation embodiments.

FIG. **3**B depicts a flowchart for describing a preferred embodiment of the whereabouts update event of a triangulated MS, for example DLM **200**, when MS location awareness is monitored by some remote service. While FIG. **3**A location determination with TDOA and AOA is well known in the art, FIGS. **3**B and **3**C include relevant processing for MSs to maintain their own whereabouts. Processing begins at block **310** and continues to block **312** where base stations able to communicate to any degree with a MS continue reporting to their controller the MS signal strength with an MS identifier (i.e. a unique handle) and Time Difference of Arrival (TDOA) information, Angle of Arrival (AOA) information, or heterogeneously both TDOA and AOA (i.e. MPT), depending on the embodiment. The MS can pick signals from base stations. In some embodiments, the MS monitors a paging channel, called a forward channel. There can be multiple forward channels. A forward channel is the transmission frequency from the base tower to the MS. Either the MS provides broadcast heartbeats (FIG. **13**A) for base stations, or the base stations provide heartbeats (FIG. **13**C) for a response from the MS, or usual MS use protocol signals are detected and used (incorporating CK **1304** in usual data **1302** by MS, or CK **1314** in “usual data” **1312** by service). Usual data is the usual communications traffic data in carrying out other character **32** processing. Communication from the MS to the base tower is on what is called the reverse channel. Forward channels and reverse channel are used to perform call setup for a created session channel.

TDOA is calculated from the time it takes for a communication to occur from the MS back to the MS via the base tower, or alternatively, from a base tower back to that base tower via the MS. NTP may also be used for time calculations in a unidirectional broadcast from a base tower (FIG. **13**C) to the MS, or from the MS (FIG. **13**A) to a base tower (as described above). AOA is performed through calculations of the angle by which a signal from the MS encounters the antenna. Triangle geometry is then used to calculate a location. The AOA antenna is typically of a phased array type.

See “Missing Part Triangulation (MPT)” section below with discussions for FIGS. **11**A through **11**E for details on heterogeneously locating the MS using both TDOA and AOA (i.e. Missing Part Triangulation (MPT)). Just as high school taught geometry for solving missing parts of a triangle, so to does MPT triangulate an MS location. Think of the length of a side of a triangle as a TDOA measurement—i.e. length of time, translatable to a distance. Think of the AOA of a signal to an antenna as one of the angles of a triangle vertice. Solving with MPT analogously uses geometric and trigonometric formulas to solve the triangulation, albeit at fast processing speeds.

Thereafter, if the MS is determined to be legitimate and deserving of processing (similar to above), then block **314** continues to block **316**. If block **314** determines the MS is not participating with the service, in which case block **312** did little to process it, then processing continues back to block **312** to continue working on behalf of legitimate participating MSs. The controller at block **316** may communicate with other controllers when base stations in other cellular clusters are picking up a signal, for example, when the MS roams. In any case, at block **316**, the controller(s) determines the strongest signal base stations needed for locating the MS, at block **316**. The strongest signals that can accomplish whereabouts information of the MS are used. Thereafter, block **318** accesses base station location information for base stations determined at block **316**. The base station provides stationary references used to (relatively) determine the location of the MS. Then, block **320** uses the TDOA, or AOA, or MPT (i.e. heterogeneously both AOA and TDOA) information together with known base station locations to calculate the MS location.

Thereafter, block **322** accesses historical MS location information, and block **324** performs housekeeping by pruning location history data for the MS by time, number of entries, or other criteria. Block **326** then determines a heading (direction) of the MS based on previous location information. Block **326** may perform Artificial Intelligence (AI) to determine where the MS may be going by consulting many or all of the location history data. Thereafter, block **328** completes a service side WDR **1100**, block **330** appends the WDR information to location history data and notifies a supervisory service if there is one outside of the service processing of FIG. **3**B. Processing continues to block **332** where the service communicates the WDR to the located MS.

Thereafter, the MS completes its own WDR at block **334** for adding to WDR queue **22** to know its own whereabouts whenever possible, and block **336** prepares parameters for invoking WDR insertion processing at block **338**. Parameters are set for: WDRREF=a reference or pointer to the MS WDR; DELETEQ=FIG. **3**B location queue discard processing; and SUPER=FIG. **3**B supervisory notification processing (e.g. no supervisory notification processing because it was already handled at block **330**, or by being in context of the FIG. **3**B service processing). At block **338**, the MS invokes FIG. **2**F processing already described. After block **338**, processing continues back to block **312**. Of course, block **332** continues directly to block **312** at the service(s) since there is no need to wait for MS(s) processing in blocks **334** through **338**. FIG. **3**B processing is continuous for every MS in the wireless network 7 days a week, 24 hours a day.

See FIG. **11**A descriptions. Fields are set to the following upon exit from block **334**:

MS ID field **1100***a* is preferably set with: Same as was described for FIG. **2**D (block **236**) above.

DATE/TIME STAMP field **1100***b* is preferably set with: Same as was described for FIG. **2**D (block **236**) above.

LOCATION field **1100***c* is preferably set with: The triangulated location of the MS as communicated by the service.

CONFIDENCE field **1100***d* is preferably set with: Confidence of triangulation determined by the service which is passed to the MS at block **332**. The confidence value may be set with the same value (e.g. 85) regardless of how the MS was triangulated. In other embodiments, field **1100***d* will be determined (completely, or adjusting the value of 85) by the service for TDOA measurements used, AOA measurements, signal strengths, wave spectrum involved, and/or the abundance of particular MS signals available for processing by blocks **312** through **320**. Higher confidences are assigned for smaller TDOA measurements (shorter distances), strong signal strengths, and numerous additional data points beyond what is necessary to locate the MS. Lower confidences are assigned for larger TDOA measurements, weak signal strengths, and minimal data points necessary to locate the MS. A reasonable confidence can be assigned using this information as guidelines where 1 is the lowest confidence and 100 is the highest confidence.

LOCATION TECHNOLOGY field **1100***e* is preferably set with: “Server Cell TDOA”, “Server Cell AOA”, “Server Cell MPT”, “Server Antenna TDOA”, “Server Antenna AOA”, or “Server Antenna MPT”, depending on how the MS was located and what flavor of service was used. The originator indicator is set to DLM.

LOCATION REFERENCE INFO field **1100***f* is preferably set with: null (not set) for indicating that all triangulation data was factored into determining confidence, and none is relevant for a single TDOA or AOA measurement in subsequent processing (i.e. service did all the work).

COMMUNICATIONS REFERENCE INFO field **1100***g* is preferably set with: Same as was described for FIG. **2**D (block **236**) above.

SPEED field **1100***h* is preferably set with: Service WDR information at block **332**, wherein the service used historical information and artificial intelligence interrogation of MS travels to determine, if available.

HEADING field **1100***i* is preferably set with: Service WDR information at block **332**, wherein the service used historical information and artificial intelligence interrogation of MS travels to determine, if available.

ELEVATION field **1100***j* is preferably set with: Elevation/altitude, if available.

APPLICATION FIELDS field **1100***k* is preferably set with: Same as was described for FIG. **2**D (block **236**) above.

CORRELATION FIELD **1100***m* is preferably set with: Not Applicable (i.e. not maintained to queue **22**).

SENT DATE/TIME STAMP field **1100***n* is preferably set with: Not Applicable (i.e. not maintained to queue **22**).

RECEIVED DATE/TIME STAMP field **1100***p* is preferably set with: Not Applicable (i.e. not maintained to queue **22**).

FIG. **3**C depicts a flowchart for describing a preferred embodiment of the whereabouts update event of a triangulated MS, for example a DLM **200**, when MS location awareness is monitored by the MS. Communications between the base stations and MS is similar to FIG. **3**B processing except the MS receives information (FIG. **13**C) for performing calculations and related processing. Processing begins at block **350** and continues to block **352** where the MS continues receiving (FIG. **13**C) pulse reporting from base stations (or antennas). AOA, TDOA, and MPT (See “Missing Part Triangulation (MPT)” section below with discussions for FIGS. **11**A through **11**E for details on heterogeneously locating the MS using both TDOA and AOA) can be used to locate the MS, so there are many possible signal types received at block **352**. Then, block **354** determines the strongest signals which can accomplish a completed WDR, or at least a location, of the MS. Thereafter, block **356** parses base station location information from the pulse messages that are received by the MS. Block **358** communicates with base stations to perform TDOA and/or AOA measurements and calculations. The time it takes for a communication to occur from the MS back to the MS for TDOA, or alternatively, from a base tower back to that base tower can be used. NTP may also be used, as described above, so that base towers (or antennas) broadcast signals (FIG. **13**C) picked up by the MS which already contain the base tower locations and NTP date/time stamps for TDOA calculations. Block **358** uses the TDOA and/or AOA information with the known base station information to determine the MS location. While AOA information from the base stations (or antennas) is used by the MS, various MS embodiments can use AOA information detected at an MS antenna provided the heading, yaw, pitch, and roll is known at the MS during the same time as signal reception by the MS. A 3-axis accelerometer (e.g. in iPhone) may also provide yaw, pitch and roll means for proper AOA calculation.

Thereafter, block **360** accesses historical MS location information (e.g. WDR queue **22** and/or LBX history **30**) to prevent redundant information kept at the MS, and block **362** performs housekeeping by pruning the LBX history **30** for the MS by time, number of entries, or other criteria. Block **364** then determines a heading (direction) of the MS based on previous location information (unless already known from block **358** for AOA determination). Block **364** may perform Artificial Intelligence (AI) to determine where